Redbooks
ibm.com/redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM watsonx.data

Deepak Rangarao
Daniele Comi
Gopi Varadarajulu
Jun Liu
Karen Medhat
Malcolm Singh
Mark Simmonds
Payal Patel
Prabh Matharu
Saurabh Kaushik
Sreenath Devireddy
Ugur Ozker
Vasfi Gucer

Artificial Intelligence

Data and AI

IBM

Redbooks

IBM Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM watsonx.data

January 2025

SG24-8570-00

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

First Edition (January 2025)

This edition applies to IBM watsonx.data Version 2.0.x.

Contents

Notices

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

DataStage®
DB2®
Db2®
IBM®
IBM Cloud®
IBM Cloud Pak®
IBM Research®
Netezza®
Redbooks®
Redbooks (logo) ®
Resilient®
Think®

The following terms are trademarks of other companies:

ITIL is a Registered Trade Mark of AXELOS Limited.

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

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

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

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

Foreword

This trilogy of IBM® Redbooks® publications positions and explains IBM watsonx - a portfolio of AI products that accelerates the impact of generative AI in core workflows to drive productivity. Each book focuses on one of the three main components of the watsonx platform:

- **IBM watsonx.ai:** A next-generation enterprise studio for AI builders to train, validate, tune, and deploy both traditional ML and new generative AI capabilities powered by foundation models.
- **IBM watsonx.data:** The hybrid, open data lakehouse to power AI and analytics with all your data, anywhere.
- **IBM watsonx.governance:** A set of AI governance capabilities enabling trusted AI workflows, helping organizations implement and comply with ever-changing industry and government regulations.

Organizations have long recognized the value that IBM Redbooks provide in guiding them with best practices, frameworks, clear explanations, and use cases as part of their solution evaluations and implementations.

This trilogy of books was only possible due to the close collaboration involving many skilled and talented authors that were selected from our IBM global technical sales, development, Expert Labs, Client Success Management, and consulting services organizations, using their diverse skills, experiences, and technical knowledge across the watsonx platform.

I would like to thank the authors, contributors, reviewers, and the IBM Redbooks team for their dedication, time, and effort in making this publication a valuable asset that organizations can use as part of their journey to AI.

I also want to thank Mark Simmonds and Deepak Rangarao for taking the lead in shaping this request into yet another successful IBM Redbooks project.

It is my sincere hope that you enjoy this watsonx trilogy as much as the team who wrote and contributed to them.

Steve Astorino, IBM General Manager - Development, Data, AI and Sustainability

Preface

IBM® watsonx™ is a portfolio of AI products that accelerates the impact of generative AI in core workflows to drive productivity. This book focuses on watsonx.data, one of the three main components of the platform.

IBM watsonx.data is a hybrid, open data lakehouse to power AI and analytics with all your data, anywhere. Built on an open lakehouse architecture, it is optimized for governing data and AI workloads. Its capabilities include querying, governance, and support for open data formats, enabling easy data access and sharing.

The solution can manage workloads both on premises and across hybrid multi-cloud environments while leveraging internal and external data sets. Through workload optimization, with this solution an organization can reduce data warehouse costs by up to 50 percent. It enables users to access robust data through a single point of entry while applying multiple fit-for-purpose query engines to uncover valuable insights. It also provides built-in data governance tools, automation, and integration with an organization's existing databases and tools to simplify setup and the user experience.

This IBM Redbooks® publication provides a broad understanding of watsonx.data concepts and architecture, and the services that are available in the product. In addition, several common use cases and scenarios are included that should help you better understand the capabilities of this product.

This publication is for watsonx customers who seek best practices and real-world examples of how to best implement their solutions while optimizing the value of their existing and future technology, AI, data, and skills investments.

Note: Other books in this series are:

- *Simplify Your AI Journey: Ensuring Trustworthy AI with IBM watsonx.governance*, SG24-8573
- *Simplify Your AI Journey: Unleashing the Power of AI with IBM watsonx.ai,* SG24-8574

Authors

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

Deepak Rangarao is an IBM Distinguished Engineer and CTO responsible for Technical Sales-Cloud Paks. Currently, he leads the technical sales team to help organizations modernize their technology landscape with IBM Cloud® Paks. He has broad cross-industry experience in the data warehousing and analytics space, building analytic applications at large organizations and technical pre-sales with start-ups and large enterprise software vendors. Deepak has co-authored several books on topics, such as OLAP analytics, change data capture, data warehousing, and object storage and is a regular speaker at technical conferences. He is a certified technical specialist in Red Hat OpenShift, Apache Spark, Microsoft SQL Server, and web development technologies.

Daniele Comi is a Data Scientist, AI Engineer, and Software Engineer at IBM Italy, with over three years of experience in data analytics, machine learning (ML), and deep learning. His expertise spans the entire spectrum of AI, from architectural design to scientific research, with a focus on machine, reinforcement, and deep learning. Daniele holds a Master's degree in Computer Science Engineering, specializing in AI frameworks and models. At IBM, Daniele

has been a key member of the AI and Generative AI team in Italy, where he has designed and implemented complex AI and generative AI architectures for a variety of industry applications. His technical expertise also includes Fully Homomorphic Encrypted AI, enabling secure AI solutions that ensure data privacy.
Daniele has collaborated with IBM Research®, authoring extensive research publications on advanced deep learning techniques, particularly encoder-decoder models. He has also contributed to IBM intellectual property through the submission of patents and the authorship of IBM Redbooks.

Gopi Varadarajulu is a Senior Technical Staff Member & Architect in the IBM Data and AI portfolio. He has an insider view of design & development of various IBM Data & AI technologies such as watsonx.data, watsonxd.ai, Watson Studio, Data Virtualization, IBM Knowledge Catalog, Watson Machine Learning & IBM Cloud Pak for Data components. He has held various design & engineering responsibilities for the past 20 years and has 6 patents in the areas of Privacy, Secure Data Analysis, Database management systems and operating systems.

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

Karen Medhat is a Customer Success Manager Architect in the UK and the youngest IBM Certified Thought Leader Level 3 Technical Specialist. She is the Chair of the IBM Technical Consultancy Group and an IBM Academy of technology member. She holds an MSc degree with honors in Engineering in AI and Wireless Sensor Networks from the Faculty of Engineering, Cairo University, and a BSc degree with honors in Engineering from the same faculty. She co-creates curriculum and exams for different IBM professional certificates. She also created and co-created courses for IBM Skills Academy in various areas of IBM technologies. She serves on the review board of international conferences and journals in AI and wireless communication. She also is an IBM Inventor and experienced in creating applications architecture and leading teams of different scales to deliver customers' projects successfully. She frequently mentors IT professionals to help them define their career goals, learn new technical skills, or acquire professional certifications. She has authored publications on Cloud, IoT, AI, wireless networks, microservices architecture, and Blockchain.

Malcolm Singh is a Technical Product Manager in the Data and AI division within IBM Software, focusing on technical strategy and connectivity. Previously, he was a Solution Architect for IBM Technology Expert Labs in the Data and AI Platforms Team. As a Solution Architect in the Expert Labs, he worked with top IBM clients worldwide, including Fortune 500 and Global 500 companies, providing guidance and technical assistance for their Data and AI Platform enterprise environments. Malcolm is based at the IBM Canada Lab in Toronto and holds a Bachelor of Science Honours degree in Computer Science from McMaster University.

Mark Simmonds is a Program Director in IBM Data and AI. He writes extensively on AI, data science, and data fabric, and holds multiple author recognition awards. He previously worked as an IT architect leading complex infrastructure design and corporate technical architecture projects. He is a member of the British Computer Society, holds a Bachelor's Degree in Computer Science, is a published author, and a prolific public speaker.

Payal Patel works in Data & AI Technical Content Development at IBM, creating technical learning materials for sellers, business partners, and clients to enable them to get the most value out of IBM's Data & AI products and solutions. She's worked in various roles at

IBM including marketing analytics, and as a Solutions Architect in IBM Technology Expert Labs, with a focus on Data & AI.

She has worked in various technical roles across the financial services, insurance, and technology industries. She holds a Bachelor of Science in Information Science from UNC Chapel Hill, and a Masters in Analytics from North Carolina State University.

Prabh Matharu is a Technical Data Architect in IBM Expert Labs, with a focus on IBM Cloud Pak for Data and Data Fabric solutions. He has delivered and worked with cross-industry customers across the EMEA region. He has developed numerous IBM Cloud Pak for Data professional certifications. He holds a Bachelor of Science degree in Mathematics.

Saurabh Kaushik serves as Program Director within watsonx.data Product Management. His focus is on building a truly open Lakehouse platform that empowers customers to leverage their data seamlessly. With over 20 years of experience in building winning products and platforms across Data, Digital, Cloud, and AI, he brings a wealth of knowledge from both enterprise and tech startup environments, catering to diverse industries and functions.

Sreenath Devireddy works as a backend developer on the Watsonx.data console team at IBM. Having recently joined IBM, he quickly developed an interest in Milvus and its transformative use cases in AI. Before joining IBM, Sreenath gained experience working with early-stage startups in the health and conversational AI domains. He holds a Master’s degree in Computer Science from the University of Cincinnati.

Ugur Ozker is the Senior Solution Architect for IBM watsonx services in the MEA region. In this role, he specializes in crafting tailored solutions for clients. His expertise lies in optimizing the integration of responsible AI, data services, and other technology products with third-party or IBM services to seamlessly align with clients' internal operations. With over 13 years of experience across diverse technology domains, he has witnessed firsthand the transformative power of technological advancements. Before joining IBM, he held a key role in the banking and finance sector, serving as a technical officer responsible for infrastructure and architecture. His tenure at the country's largest public bank provided him with invaluable insights into the intricacies of digital transformation.

Vasfi Gucer leads projects for the IBM Redbooks team, leveraging his 20+ years of experience in systems management, networking, and software. A prolific writer and global IBM instructor, his focus has shifted to storage and cloud computing in the past eight years. Vasfi holds multiple certifications, including IBM Certified Senior IT Specialist, PMP, ITIL V2 Manager, and ITIL V3 Expert.

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

- Steve Astorino, IBM General Manager - Development, Data, AI and Sustainability
- Jamie Roszel, IBM Redbooks Content Developer, **IBM RTP**

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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

1

Challenges and opportunities with data

A long-standing challenge many organizations face is ensuring their data assets are accessible, manageable, governed, and of high quality for use in new (artificial intelligence) AI applications. These applications integrate AI throughout the enterprise for smarter business outcomes.

Over the years, numerous paradigms and efforts have attempted to address the complexities of managing sprawling and disparate data silos that all fell short of their promises and expectations. Organizations also need to place their data and assets where it makes the most business sense - whether on premises, in private or public clouds, or a combination thereof - without detriment to their business operations for everyone's benefit.

As the volume, velocity, and variety of data continue to grow rapidly, organizations must invest in robust infrastructure and analytics tools to effectively manage and extract value from their vast data sets.

This chapter describes the challenges and opportunities with data:

1.1 Current challenges in the data landscape

Accurate and accessible data, in all its forms (structured, unstructured, multimedia, digital, genetic, and organic) is the lifeblood of an organization. With the correct data, artificial intelligence (AI) can help an organization accelerate the achievement of smarter outcomes through deeper insights and understanding of its data estates, and discover innovative ways of solving some of the world’s most challenging business and societal problems.

Yet many organizations continue to face the challenges of managing, governing, and analyzing sprawling disparate data silos spread across their multi-vendor on-premises, private cloud, or public cloud, and hybrid multi-cloud environments.

Many new paradigms and advances in computing technologies have sought to revolutionize the ways in which organizations analyze and extract insights from data. Over time, consumers learn the values and limitations of these technologies. Some technologies have a short lifespan. Others endure and evolve over decades and remain relevant to this day, such as relational database management systems (RDBMSs).

1.1.1 From centralized to distributed

For many years, data storage and processing were centralized. People had to take their work to the computer or access it through “dumb” terminals. With the advent of more affordable computers, processing and data became decentralized, putting computing power in the hands of individuals. However, this development led to a problem of data being replicated in an uncontrolled manner.

With data being created, stored, and processed across many personal devices, it became increasingly difficult to control the sprawl of versions of data sets and apply quality, security, and other controls. Individual departments in various enterprises started organizing and storing only the data that they needed, which resulted in many data silos that did not communicate with each other across an organization.

1.1.2 Data stores, data integration, and data management tools

Numerous solutions appeared for managing and integrating data to enable reporting, analysis, and discovery of insights as data volumes grew. They were data stores with names like database, online transactional processing (OLTP), online analytical processing (OLAP), data warehouse, MDM system, data mart, data lake, Hadoop, or data lakehouse. Many of these terms tend to be used interchangeably, but important differences exist. Each term provides certain capabilities and values to different groups of users, but none is a panacea for all data management challenges. However, technology follows a maturity curve or cycle, and these technologies eventually found their own niches as they matured.

Many forms of data stores and data servers are used across enterprises today. More variations of them and new paradigms will evolve because technology constantly advances.

A *data fabric* (an architectural approach that simplifies data access in an organization and facilitates self-service data consumption) can offer enough longevity and flexibility to integrate an organization’s current and future data assets and enable them for AI applications. This subject alone warrants a book of its own.

1.1.3 Data lakes and data warehouses

Much of an organization's data is often distributed across many disparate silos, making it difficult to integrate and access. Data is represented in many different and sometimes complex formats as the market supports new paradigms and more diverse use cases that use unstructured and semi-structured data. Much time is spent transforming, cleansing, and integrating data. Data lakes represent a way to store massive amounts of different data by using cheap, commoditized hardware and storage through Hadoop, HDFS, or Hive. However, an organization's enthusiasm might decrease if they find it difficult to manage and get a good return on their data lake investment or even rely on the data (partly due to poor-quality data). Poor-quality data continues to concern most organizations. Once poor-quality data is shared across multiple business units and decisions are made based on that data, it can be a difficult, costly, and lengthy process to recover from that mistake.

Data warehouses can be considered as the first iteration of tools to support analytics, which enabled organizations to analyze data and decide at scale. Data warehouses enabled organizations to look at historical data. Over time, as volumes continued to grow, it became challenging to store and retain all data in a data warehouse. Also, organizations might have only a few years' worth of data or only a small slice of the operational data in their warehouses.

Organizations recognize that large volumes of unstructured data, even if they are not suitable for data warehouses, also contain great value because the data can be extracted. Unfortunately, when you try to analyze data of poor quality, the tools and analytical engines that you use to analyze the content in the data lakes might not be as performant as the tools that you use for data warehouses. Many data lakes became more like data swamps with stale data that is difficult to maintain and untrustworthy. The need to scale compute and storage presents two different sets of needs across data lakes and data warehouses.

Data lakes and data warehouses each provide their own set of capabilities. When combined, scaling and governance can become key challenges because data lakes and warehouses are designed for different purposes. The market has evolved toward cloud-based data warehouses, which offer separation of computing and storage. Technologies such as Red Hat OpenShift, Red Hat Ceph Storage, Amazon S3, and other warehouse engines help solve the problem, making storage and computing inexpensive, readily available, and simpler to manage and scale. Compute and storage must be elastic, and able to scale on demand when needed so that organizations are charged only for what they use over the billable period.

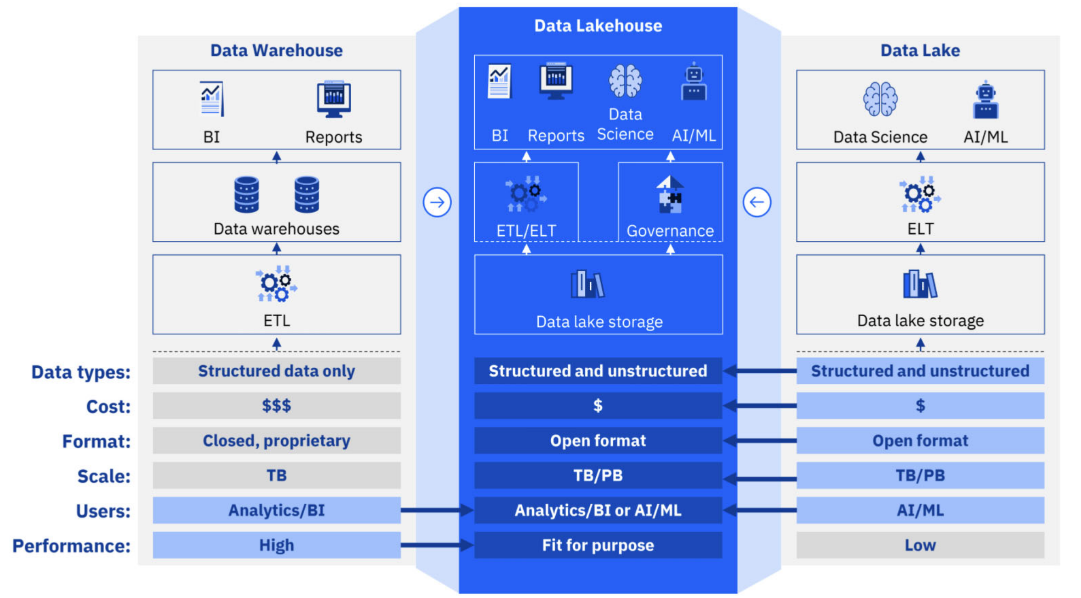
1.2 Benefits of an open lakehouse for businesses

Although data warehouses and data lakes each evolved to meet a set of specific technology and business needs and values, organizations often need both, so there is an increasing demand for convergence of both technologies. Vendors attempt to create the best of both data lakes and data warehouses by combining them into the newer technology of the lakehouse. This *lakehouse* architecture is designed to provide the flexibility and cost effectiveness of a data lake with the performance and structure of a data warehouse. The lakehouse enables organizations to store data from the increasing number of new sources in a low-cost way and use built-in data management and governance capabilities, which enable organizations to power both Business Intelligence (BI) and high-performance ML workloads efficiently and effectively.

If implemented correctly, organizations can use their investments in data lakes and data warehouses by adopting and implementing a lakehouse architecture and technologies to help modernize their existing data lakes. Enterprises can also complement their data warehouses to support some of these new types of AI- and ML-driven workloads.

A lakehouse, as shown in Figure 1-1, attempts to bridge data warehouses and data lakes by combining the best of both into one architecture. That said, these first-generation lakehouses have constraints that limit their ability to address cost and complexity challenges:

- Single-query engines are set up to support limited workloads, typically just for BI and ML.
- Lakehouses are typically deployed only over the cloud, with no support for hybrid multi-cloud deployments.
- Lakehouses offer minimal governance and metadata capabilities to deploy across an entire ecosystem.



	Data Warehouse	Data Lakehouse	Data Lake
Data types:	Structured data only	Structured and unstructured	Structured and unstructured
Cost:	$$$	$	$
Format:	Closed, proprietary	Open format	Open format
Scale:	TB	TB/PB	TB/PB
Users:	Analytics/BI	Analytics/BI or AI/ML	AI/ML
Performance:	High	Fit for purpose	Low

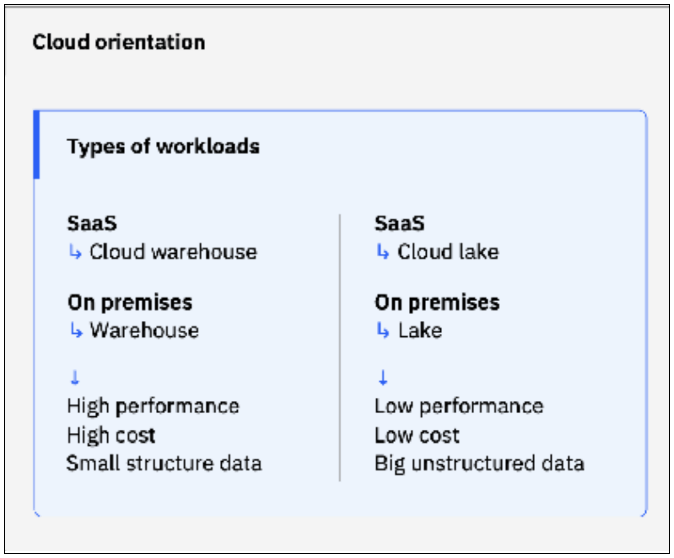
Figure 1-1 Lakehouses combine the best of data warehouses and data lakes

Data storage paradigms are cumulative. They do not disappear or are replaced by the next paradigm. They must coexist. As these technologies mature, organizations recognize the value that each paradigm provides at performing certain tasks.

Every step of an organization’s data or AI journey is critical. AI is not magic; it requires a thoughtful and well-designed approach. For example, most AI failures are due to problems in data preparation and data organization, and not the AI models themselves. Success with AI models depends on achieving success first with how the data is collected, stored, organized, and managed.

1.2.1 The impact of cloud

A combination of on-premises and cloud-native warehouses and custom data lakes is common for an enterprise architecture, as shown in Figure 1-2. Juggling costs, siloed data, and data governance are constant challenges.



Cloud orientation

Types of workloads

SaaS
↳ Cloud warehouse

On premises
↳ Warehouse

↓
High performance
High cost
Small structure data

SaaS
↳ Cloud lake

On premises
↳ Lake

↓
Low performance
Low cost
Big unstructured data

Figure 1-2 On-premises, cloud-native warehouses, and custom data lakes are commonly found in enterprise architectures

Most lakehouse solutions offer a high-performance query engine over low-cost storage with a metadata governance layer. Intelligent metadata layers enable users to categorize and classify unstructured data (such as video and voice) and semi-structured data (such as eXtensible Markup Language (XML), JavaScript Object Notation (JSON), and emails). In an ideal world, a lakehouse offers open-source technologies that reduce data duplication and simplify complex Extract, Transform, Load (ETL) pipelines. Some first-generation lakehouses have key constraints that limit their ability to address the challenges of cost and complexity. For example, a single-query engine that is designed for BI or ML workloads might be ineffective when it is used for another workload type.

Different workloads should be optimized with the best-suited environment to keep costs at a minimum and performance at a maximum. Organizations need a lakehouse that delivers an optimal level of performance for better decision-making along with the ability to unlock more value from all types of data, resulting in deeper insights.

It is an evolution of the analytic data repository that supports refinement, delivery, and storage with an open table format. Apache Iceberg is designed to handle huge analytic data sets. It is used in production environments where a single table can contain tens of petabytes of data and the data can be read without a distributed SQL engine.

1.3 Open table formats and open data formats

Adopting an open architecture when implementing a lakehouse helps ensure interoperability across today's hybrid multicloud environments, which often incorporate multiple vendors' hypervisors.

1.3.1 Open data storage

This section describes some prominent open data storage services that are on the market at the time of writing.

Amazon S3

Amazon Simple Storage Service (S3) is a service that is offered by Amazon Web Services (AWS) and provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its e-commerce network. Amazon S3 can store any type of object, which enables storage for internet applications, backups, disaster recovery, data archives, data lakes for analytics, and hybrid cloud storage. AWS started Amazon S3 in the United States on March 14, 2006, and then in Europe in November 2007.

Azure Storage

The Azure Storage platform is Microsoft's cloud storage solution for modern data storage scenarios. Azure Storage offers highly available, massively scalable, durable, and secure storage for various data objects in the cloud. Azure Storage data objects are accessible from anywhere in the world over HTTP or HTTPS through a REST API. Azure Storage also offers client libraries for developers building applications or services with .NET, Java, Python, JavaScript, C++, and Go. Developers and IT professionals can use Azure PowerShell and Azure CLI to write scripts for data management or configuration tasks.

IBM Cloud Object Storage

IBM Cloud Object Storage is a service that is offered by IBM for storing and accessing unstructured data. This service can be deployed on-premises, as part of IBM Cloud Platform offerings, or in hybrid form. The offering can store any type of object for data archiving and backup, web and mobile applications, and as scalable, persistent storage for analytics. Interaction with IBM Cloud Object Storage is based on Rest APIs.

Red Hat Ceph Storage

Ceph is a no-charge, open-source, and software-defined storage platform that provides object storage, block storage, and file storage built on a common distributed cluster foundation. Ceph provides distributed operation without a single point of failure and scalability to the exabyte level. Since Version 12 (Luminous), Ceph does not rely on any other conventional file system and directly manages HDDs and SSDs with its own storage back end BlueStore. It can expose a POSIX file system.

Ceph replicates data with fault tolerance by using commodity hardware and Ethernet IP, and it requires no specific hardware support. Ceph is highly available and helps ensure strong data durability through techniques that include replication, erasure coding, snapshots, and clones. By design, the system is both self-healing and self-managing, minimizing administration time and other costs.

Google Cloud Storage

Google Cloud Storage is a RESTful online file storage web service for storing and accessing data on the Google Cloud Platform infrastructure. The service combines the performance and scalability of Google's cloud with advanced security and sharing capabilities. It is an Infrastructure as a Service (IaaS), and it is comparable to Amazon S3. Contrary to Google Drive and according to different service specifications, Google Cloud Storage appears to be more suitable for enterprises.

HDFS

Hadoop Distributed File System (HDFS) is a distributed file system that handles massive data sets efficiently on commodity hardware. As a core component of Apache Hadoop, HDFS enables the scaling of single clusters to hundreds or even thousands of nodes. It works in tandem with MapReduce and YARN to form the foundation of the Hadoop ecosystem.

1.3.2 Open data formats

This section describes some prominent open data formats.

Apache Iceberg

Apache Iceberg is an open-source, high-performance format for huge analytic tables. Apache Iceberg enables SQL tables for big data while making it possible for engines like Spark, Trino, Flink, Presto, Hive, Impala, StarRocks, Doris, and Pig to work with the same tables at the same time. Apache Iceberg is released under the Apache License. Apache Iceberg addresses the performance and usability challenges of using Apache Hive tables in large and demanding data lake environments. Vendors supporting Apache Iceberg tables in their products at the time of writing include Buster, CelerData, Cloudera, Dremio, IOMETE, Snowflake, Starburst, Tabular, and AWS.

Apache Parquet

Apache Parquet is a no-charge, open-source, and column-oriented data storage format in the Apache Hadoop ecosystem. It is similar to Apache Hive Record Columnar File (RCFile) and Apache Optimized Row Columnar (ORC), which are the other columnar-storage file formats in Hadoop. It is compatible with most of the data processing frameworks of Apache Hadoop. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

Apache Avro

Apache Avro is a row-oriented, remote procedure call and data serialization framework that is developed within the Apache Hadoop project. It uses JSON for defining data types and protocols, and serializes data in a compact binary format. Its primary use is in Apache Hadoop, where it provides both a serialization format for persistent data, which is a wire format for communication between Hadoop nodes and from client programs to the Hadoop services. Avro uses a schema to structure the data that is being encoded. It has two different types of schema languages: one for human editing (Avro IDL) and another that is more machine readable (based on JSON).

It is similar to Thrift and Protocol Buffers, but does not require a code-generation program when a schema changes (unless the code-generation program is wanted for statically-typed languages).

Apache Spark SQL can access Avro as a data source.

Apache ORC

Apache ORC is a no-charge, open-source, and column-oriented data storage format. It is similar to the other columnar-storage file formats that are available in the Hadoop ecosystem, such as RCFile and Apache Parquet. It is used by most of the data processing frameworks, such as Apache Spark, Apache Hive, Apache Flink, and Apache Hadoop.

In February 2013, the ORC file format was announced by Hortonworks in collaboration with Facebook. A month later, the Apache Parquet format was announced, which is developed by Cloudera and Twitter.

The Apache ORC format is supported by Amazon AWS Glue.

1.4 Storage considerations for growing data

As organizations continue to generate and collect vast amounts of data, the need for efficient and scalable storage solutions becomes more important. This section explores the storage considerations for growing data, with a focus on the challenges and opportunities that arise when working with large datasets.

1.4.1 The data growth conundrum

The rate at which data is being generated is staggering. According to a report by IDC, the global data sphere is expected to grow from 33 zettabytes (ZB) in 2018 to 175 ZB by 2025, which represents a compound annual growth rate (CAGR) of 26%.[1] This growth is driven by many factors, which include the increasing usage of IoT devices, social media, and cloud computing. As data grows, so too do the challenges that are associated with storing it. Organizations must balance the need to store large amounts of data with the need to ensure that data is accessible, secure, and compliant with regulatory requirements.

1.4.2 Storage challenges

In storing growing data, organizations face several challenges:

- Scalability: As data grows, storage systems must scale to accommodate it. This act can be a challenge, particularly for organizations with limited resources.
- Performance: As data volumes surge, storage systems must maintain peak performance to help ensure rapid and efficient data access. Cost considerations are significant, especially when organizations rely on traditional storage solutions for large-scale data storage.
- Data protection: As data grows, the risk of data loss or corruption increases. Organizations must ensure that they have adequate data protection measures in place.

[1] Source: The Digital Universe in 2025: Emerging Trends and the Future of Data (2018)

1.4.3 Storage opportunities

Although there are challenges that are associated with storing growing data, there are also opportunities. New storage technologies and architectures are emerging that are designed to handle large amounts of data:

- Object storage: Object storage is a type of storage that is designed to handle large amounts of unstructured data. It is highly scalable and can be more cost-effective than traditional storage solutions.
- Cloud storage: Cloud storage provides organizations with a flexible and scalable storage solution that can be easily expanded or contracted as needed.
- Flash Storage: Flash Storage is a type of storage that uses flash memory to store data. It is highly performant and can be used to accelerate applications that require high levels of I/O.

1.4.4 Best practices for storage

In storing growing data, there are several best practices that organizations should follow:

- Develop a data management strategy: Organizations should develop a data management strategy that accounts for their storage needs.
- Use a tiered storage architecture: Organizations should use a tiered storage architecture that includes a combination of hot, warm, and cold storage.
- Use data compression and deduplication: Organizations should use data compression and deduplication to reduce the amount of storage that is required.

In conclusion, the growth of data presents both challenges and opportunities when it comes to storage. Organizations must balance the need to store large amounts of data with the need to help ensure that data is accessible, secure, and compliant with regulatory requirements. By understanding the storage challenges and opportunities and by using best practices and technologies such as IBM watsonx.data, organizations can help ensure that they are well positioned to handle the growth of data.

2

Introduction to IBM watsonx.data

IBM watsonx.data is a powerful data lakehouse that simplifies data management and analysis. It combines the best of data warehouses and data lakes, allowing you to store and analyze any type of data. By breaking down data silos and streamlining workflows, watsonx.data empowers you to unlock valuable insights and drive innovation.

This chapter explores the key features and benefits of watsonx.data and has the following sections:

- “Introduction to the watsonx platform and its core components” on page 12
- “IBM watsonx.data overview and architecture” on page 16
- “Benefits of using watsonx.data for businesses” on page 21
- “Data pipeline considerations for open lakehouse” on page 22
- “Data pipelines integration with watsonx.data” on page 24

2.1 Introduction to the watsonx platform and its core components

The effectiveness of AI relies on high-quality data, which is essential for informed decision-making. Governance plays a critical role in ensuring data integrity and compliance with regulations. These components must operate in tandem.

AI depends on data as its life source. The data must be of the right quality and should be accessed securely by authorized personnel only. AI enables data analysis and management, and governance needs AI to continuously learn about potential threats and human behaviors, and to comply with regulatory standards and legislation.

Without AI, data governance, or quality control, an organization's effectiveness may suffer.

IBM watsonx was designed to bridge the gap between advanced AI capabilities and trusted data. This AI and data platform is designed to enable enterprises to scale and accelerate the impact of the most advanced AI with their trusted data. Organizations turning to AI today need access to a full technology stack that enables them to train, tune, and deploy AI models, including Foundation Models and ML capabilities, across their organization with trusted data, speed, and governance—all in one place and designed to run on any cloud environment.

With watsonx, users have access to the toolset, technology, infrastructure, and consulting expertise to build their own or fine-tune and adapt available AI models on their own data and deploy them at scale in a trustworthy and open environment. Competitive differentiation and unique business value will be able to be increasingly derived from how adaptable an AI model can be to an enterprise's unique data and domain knowledge.

The IBM watsonx platform consists of three unique sets of products to help address these needs, as shown in Figure 2-1.

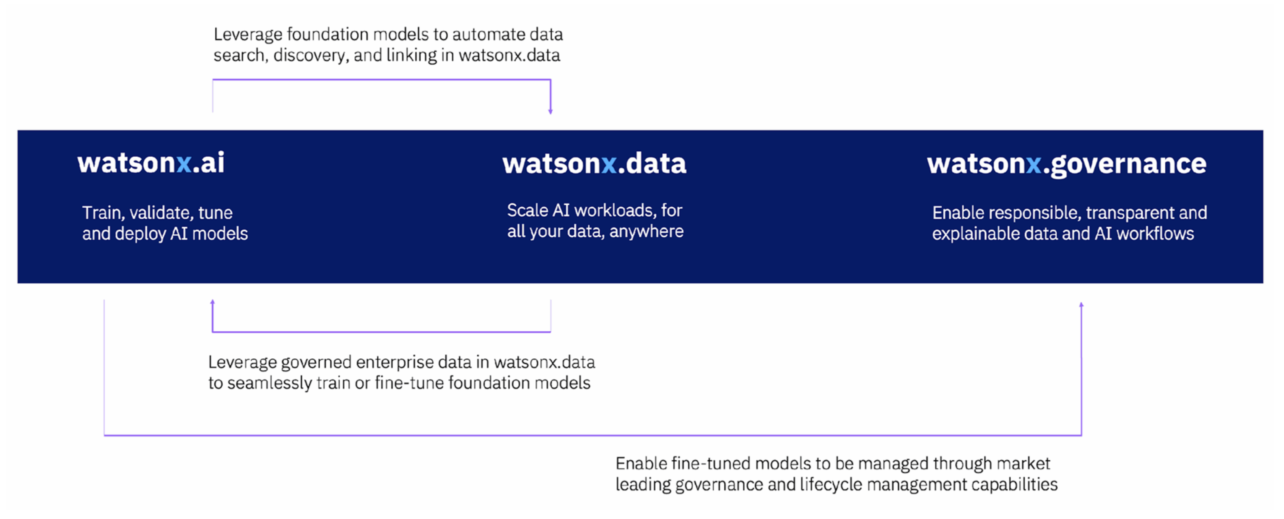

Leverage foundation models to automate data search, discovery, and linking in watsonx.data
watsonx.ai
Train, validate, tune and deploy AI models
watsonx.data
Scale AI workloads, for all your data, anywhere
watsonx.governance
Enable responsible, transparent and explainable data and AI workflows
Leverage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

Figure 2-1 Scale and accelerate the impact of AI with trusted data using IBM watsonx

As a platform, watsonx is represented in Figure 2-2 on page 13.

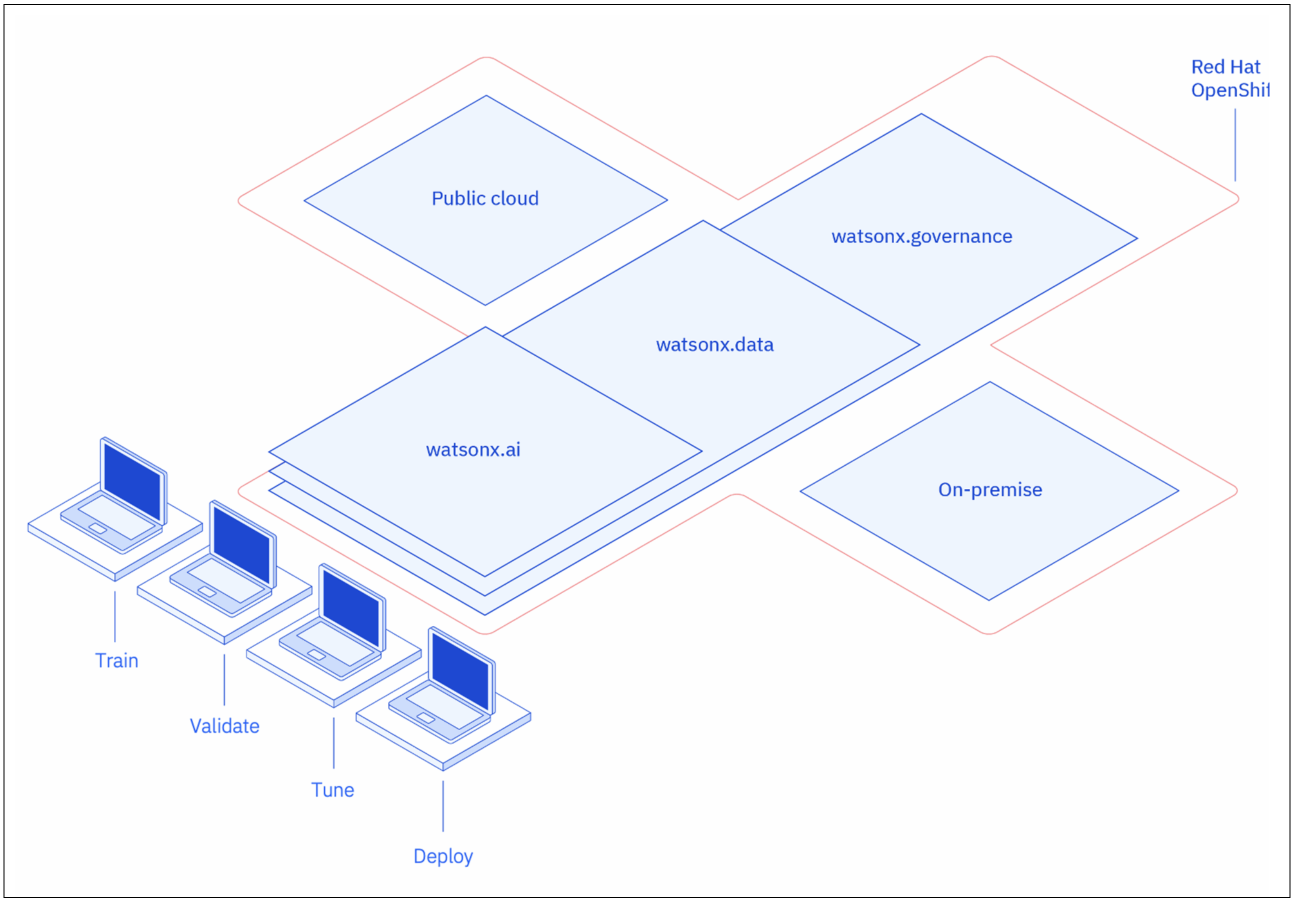


Figure 2-2 IBM watsonx conceptual architecture

2.1.1 IBM watsonx.ai

Introducing our next-generation enterprise studio for AI builders to train, validate (test), tune, and deploy both traditional ML and new generative AI capabilities powered by foundation models using an open and intuitive user interface.

The AI studio provides a range of foundation models, training and tuning tools, and a cost-effective infrastructure that facilitates the entire data and AI lifecycle, from data preparation through model development, deployment, and monitoring.

Additionally, a foundation model library that gives users easy access to IBM-curated and -trained foundation models. Foundation models use a large, curated set of enterprise data, backed by a robust filtering and cleansing process and with an auditable data lineage. These models are being trained not only on language, but on a variety of modalities, including code, time-series data, tabular data, geospatial data, and IT events data. Examples of model categories include (but not limited to):

- **fm.code:** Models built to automatically generate code for developers through a natural-language interface to boost developer productivity and enable the automation of many IT tasks
- **fm.NLP:** A collection of LLMs for specific or industry-specific domains that use curated data to help mitigate bias and more quickly make domains customizable using client data

- **fm.geospatial:** Models built on climate and remote sensing data to help organizations understand and plan for changes in natural disaster patterns, biodiversity, land use, and other geophysical processes that could impact their businesses

The watsonx.ai studio builds upon Hugging Face's open-source libraries and offers thousands of Hugging Face open models and data sets. Users can leverage the power of IBM Granite LLMs, along with the latest Mistral, Llama, and other third party LLMs. This is part of IBM's commitment to delivering an open ecosystem approach that enables users to leverage the best models and architecture for their unique business needs.

2.1.2 IBM watsonx.data

IBM watsonx.data is a powerful data lakehouse that simplifies data management and analysis. It combines the best of data warehouses and data lakes, allowing you to store and analyze any type of data. By breaking down data silos and streamlining workflows, watsonx.data empowers you to unlock valuable insights and drive innovation.

The solution can manage workloads both on premises and across hybrid multi-cloud environments while leveraging internal and external data sets.

Through workload optimization, with this solution an organization can reduce data warehouse costs by up to 50 percent. This is based on comparing published 2023 list prices normalized for virtual private cloud hours of watsonx.data to several major cloud data warehouse vendors. Savings may vary depending on configurations, workloads, and vendors.

It enables users to access robust data through a single point of entry while applying multiple fit-for-purpose query engines to uncover valuable insights.

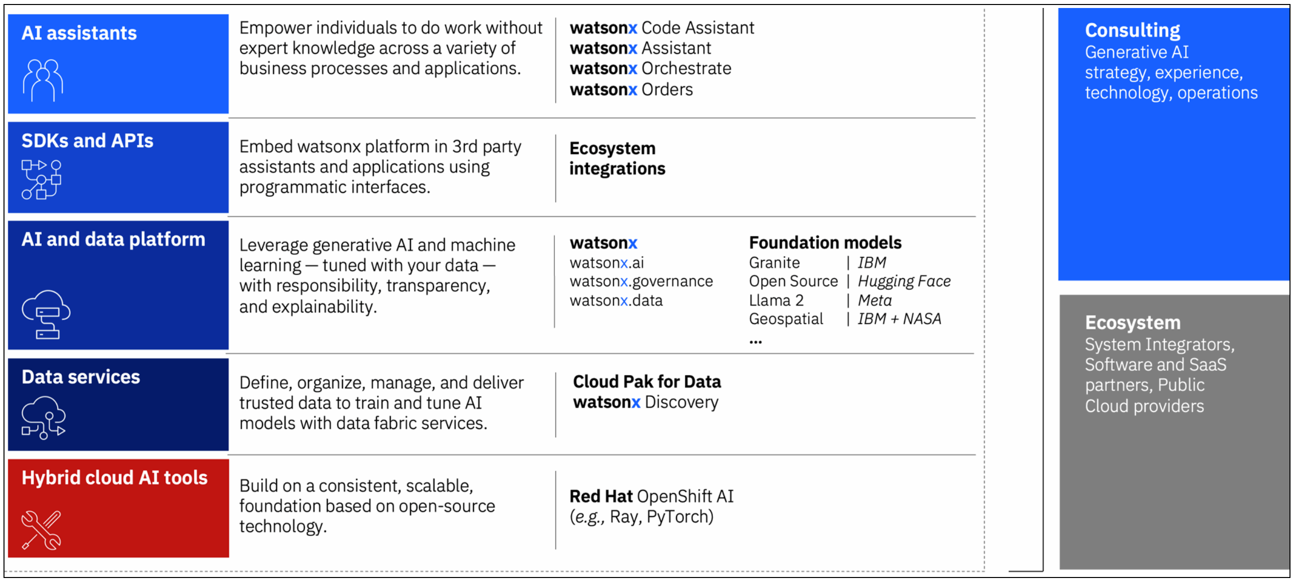
It also provides built-in governance tools, automation, and integration with an organization's existing databases and tools to simplify setup and user experience.

2.1.3 IBM watsonx.governance

This set of AI governance capabilities enables trusted AI workflows, helping organizations to:

- Operationalize governance to help mitigate the risk, time, and cost associated with manual processes. It also provides the documentation necessary to drive transparent and explainable outcomes.
- Provide the mechanisms to protect customer privacy, proactively detect model bias and drift, and meet the organization’s ethics standards.
- Meet existing and future compliance needs, such as the EU Digital Services Act and Digital Markets Act.

The IBM watsonx AI and data platform forms part of the larger IBM generative AI technology and expertise stack shown in Figure 2-3 on page 15. This stack offers organizations a complete solution no matter where they might be on their AI and data journey - from consulting and ecosystems to hybrid cloud AI tools and infrastructure, data services, the watsonx AI and data platform, SDKs and APIs, and AI assistants.



AI assistants	Empower individuals to do work without expert knowledge across a variety of business processes and applications.	**watsonx** Code Assistant **watsonx** Assistant **watsonx** Orchestrate **watsonx** Orders	
SDKs and APIs	Embed watsonx platform in 3rd party assistants and applications using programmatic interfaces.	**Ecosystem integrations**	
AI and data platform	Leverage generative AI and machine learning — tuned with your data — with responsibility, transparency, and explainability.	**watsonx** watsonx.ai watsonx.governance watsonx.data	**Foundation models** Granite | *IBM* Open Source | *Hugging Face* Llama 2 | *Meta* Geospatial | *IBM + NASA* ...
Data services	Define, organize, manage, and deliver trusted data to train and tune AI models with data fabric services.	**Cloud Pak for Data** **watsonx** Discovery	
Hybrid cloud AI tools	Build on a consistent, scalable, foundation based on open-source technology.	**Red Hat** OpenShift AI (*e.g.,* Ray, PyTorch)	

Consulting
Generative AI strategy, experience, technology, operations

Ecosystem
System Integrators, Software and SaaS partners, Public Cloud providers

Figure 2-3 IBM Generative AI technology and expertise

IBM AI technologies remain focused on being:

- Open - IBM AI is based on the best open technologies available.
- Trusted - IBM AI is transparent, responsible, and governed.
- Targeted - IBM AI is designed for enterprise and targeted at business domains.
- Empowering - IBM AI is for value creators, not just users.

2.2 IBM watsonx.data overview and architecture

IBM watsonx.data is an open, hybrid, governed data store optimized for all data, analytics, and AI workloads, conceptualized, as shown in Figure 2-4.

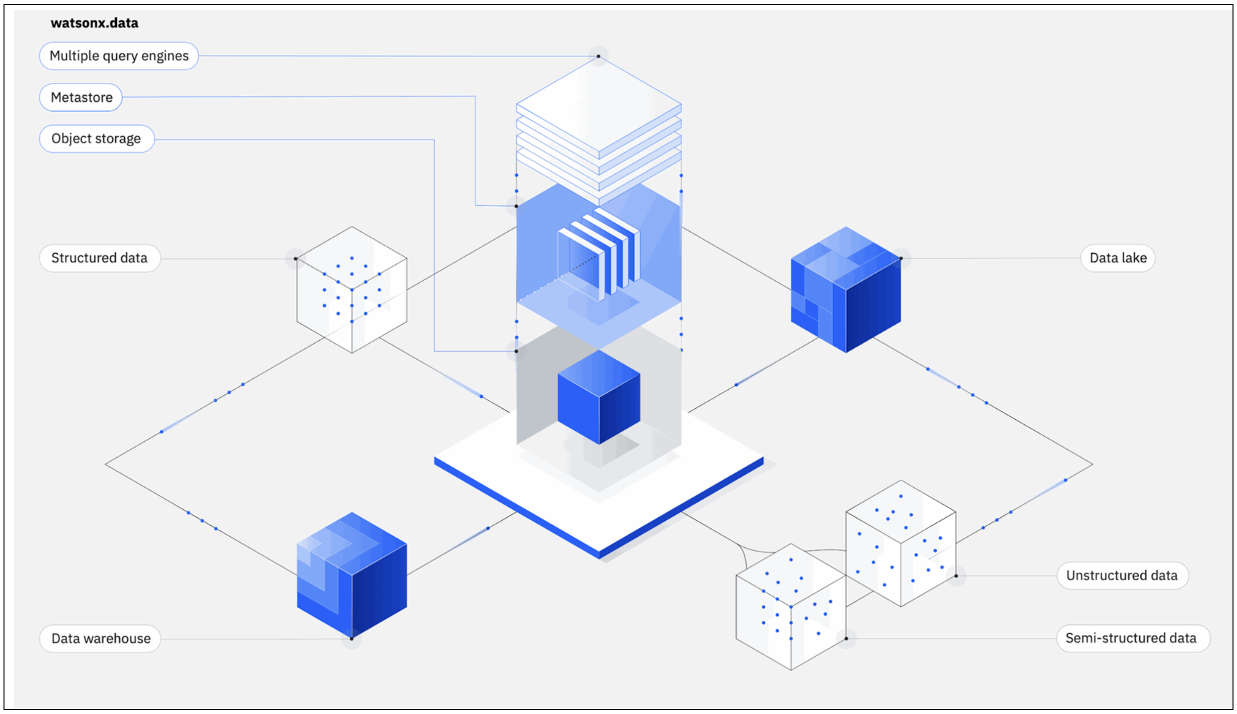


Figure 2-4 IBM watsonx.data concepts architecture

Based on our experience with numerous clients, IBM has found that organizations are often at one or more of these stages:

- Remaining on traditional warehouse or analytic appliances but looking for ways to get greater flexibility and to also perhaps tackle new workloads.
- Have adopted the traditional data lakes but are running into issues of getting sufficient return on their investment and having to manage those systems.
- Have adopted the cloud data warehouses but are concerned with ever-increasing billing costs.

All three of these groups are looking for ways to get more flexibility, adopt more workloads, reduce costs, and reduce complexity.

IBM watsonx.data is designed to address the needs of all three groups and the shortcomings of some first-generation lakehouses. It combines open, flexible, and low-cost storage of data lakes with the transactional qualities and performance of a data warehouse. This supports structured, semi-structured, and unstructured data residing in commodity storage, bringing together the best of data lakes and warehouses to enable best-in-class AI, BI, and ML in one solution without vendor lock-in.

Modularity and flexibility are key when implementing a lakehouse. If an organization has a Hadoop data lake with data stored on Hadoop Distributed File System (HDFS), the metadata can be cataloged using Hive, and the metadata and data can be brought to the lakehouse (watsonx.data) so that, from day one, the most appropriate engines can be used to query the data.

New data arriving in the lakehouse needs to be integrated with existing data using the metadata and storage layers (Hive and HDFS) and continuously analyzed without affecting existing applications using the data lake. Over time, data can be moved into the data lake at an organization's own pace.

Many of the watsonx.data components shown in Figure 2-5 are based on open-source technologies such as Presto, Iceberg, Hive, Ranger, and others. IBM watsonx.data also offers a wide range of integration with existing IBM and third-party products.

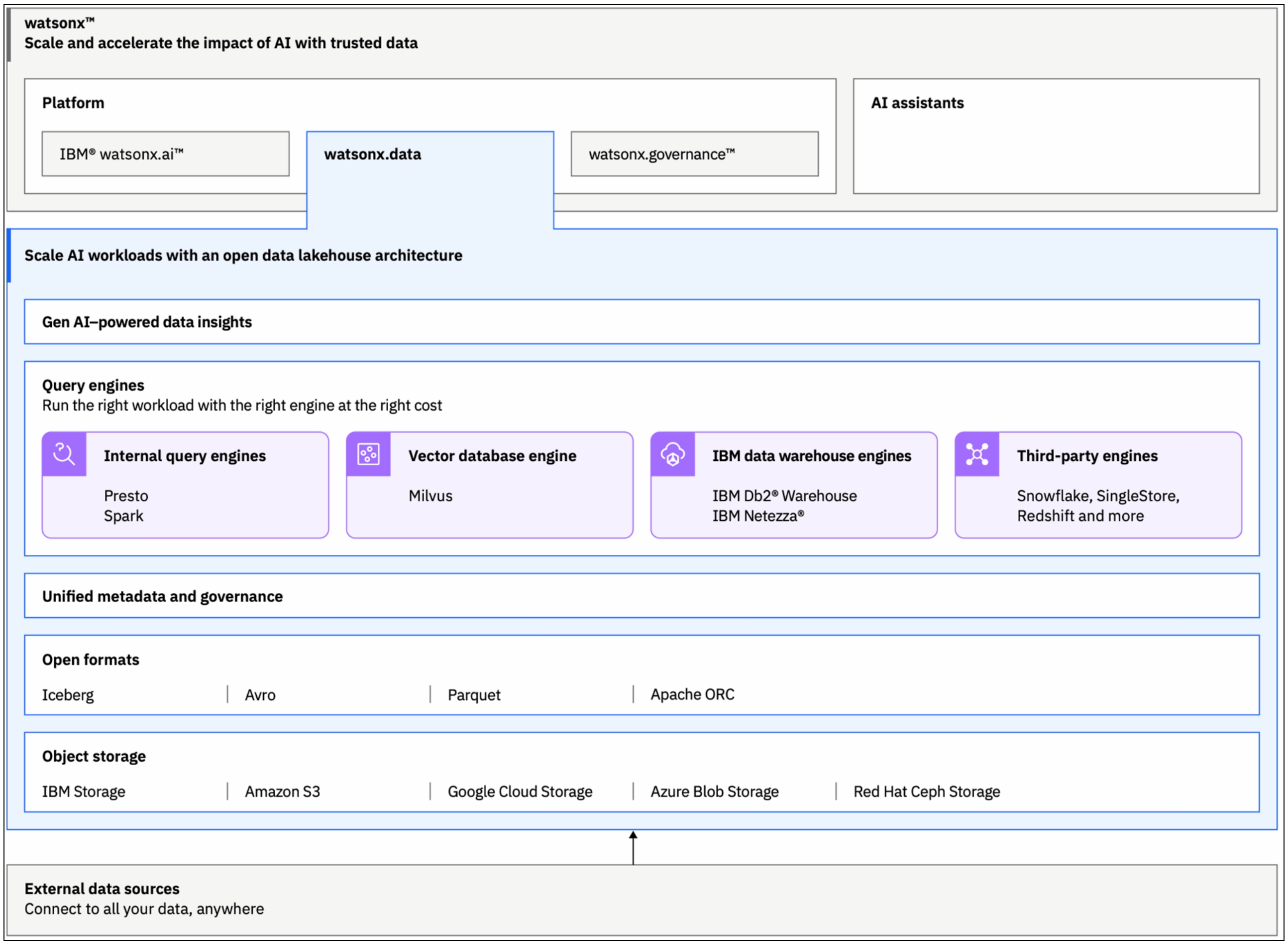


Figure 2-5 IBM watsonx.data high-level architecture

IBM watsonx.data can be deployed across multiple environments, including but not limited to IBM Cloud, Amazon Web Services (AWS) infrastructure, and on premises.

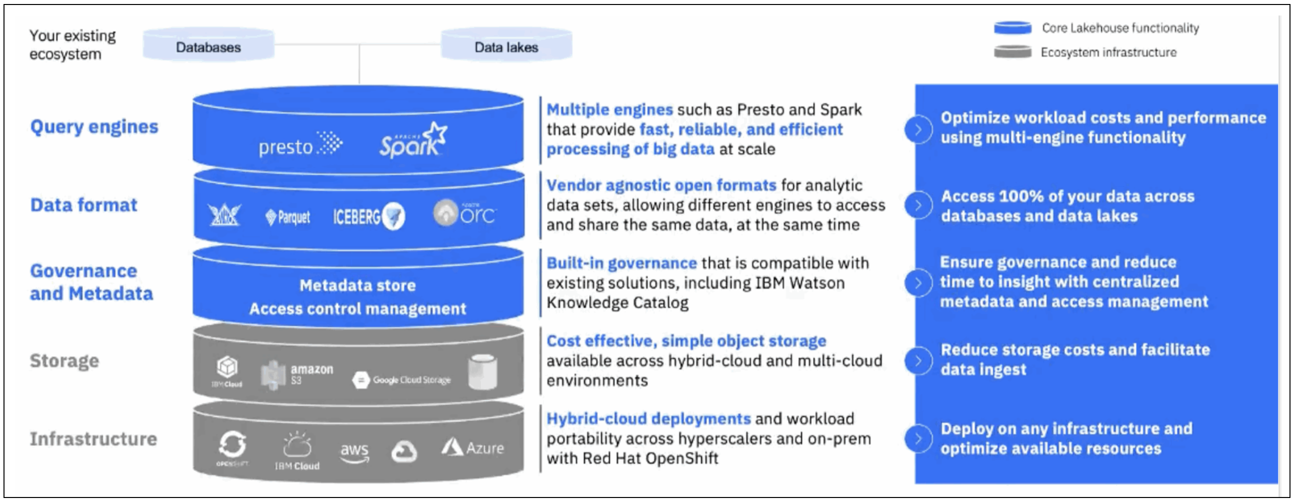
The storage layer is centered around object storage, which is highly available, highly scalable, and inexpensive.

A governance and metadata layer integrates existing Netezza® and Db2 services to achieve metadata sharing using open-data formats such as Parquet, ORC, and Avro (a serialization format for record data and for streaming data pipelines) and leverages the Apache Iceberg table format. It uses multiple engines, such as Presto and Spark, which provide fast, reliable, efficient processing of big data at scale.

With watsonx.data, you can access all your data across cloud and on-premises environments. The platform lets you connect to storage and analytics environments in minutes and access all your data through a single point of entry with a shared metadata layer.

You can use multiple query engines to optimize analytics and AI workloads for price performance and prepare your data for AI with an integrated vector database. Get greater value from your data investments with an open, hybrid, governed data lakehouse that is optimized for all data and AI workloads, and put AI to work.

Many of the watsonx.data components shown in Figure 2-6 are based on open-source technologies such as Presto, Iceberg, Hive, Ranger, and others. IBM watsonx.data also offers a wide range of integration with existing IBM and third-party products.



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

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

Built-in governance that is compatible with existing solutions, including IBM Watson Knowledge Catalog

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

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

Optimize workload costs and performance using multi-engine functionality

Access 100% of your data across databases and data lakes

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

Reduce storage costs and facilitate data ingest

Deploy on any infrastructure and optimize available resources

Figure 2-6 Overview of watsonx.data components

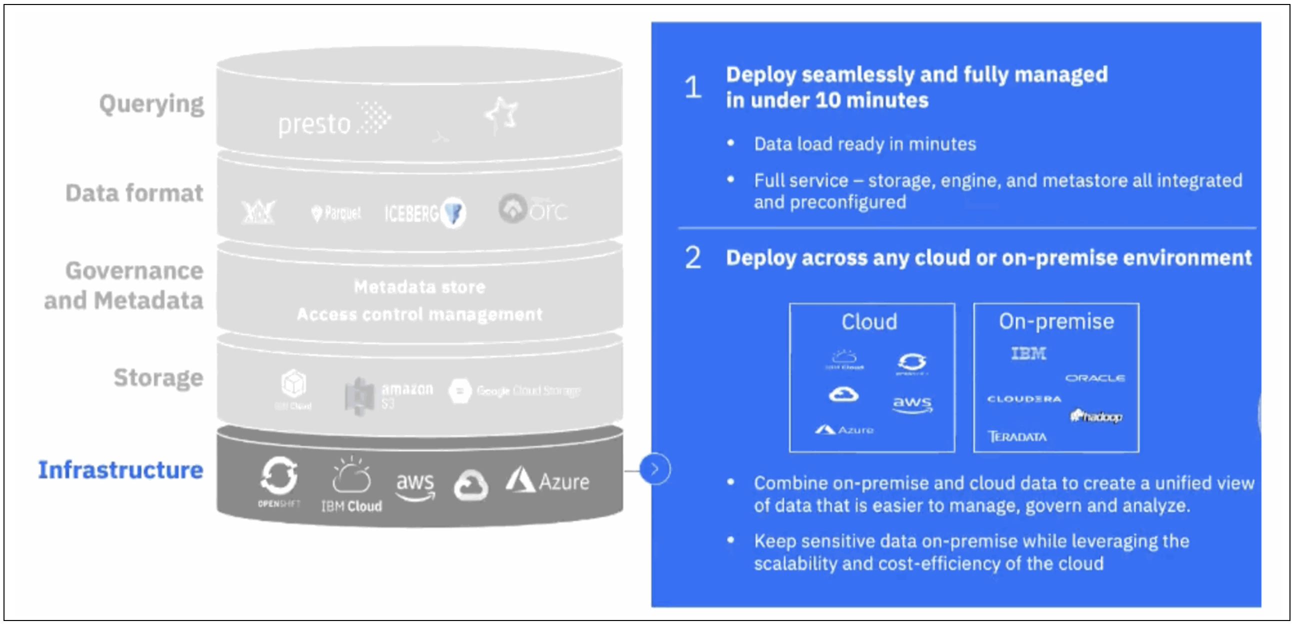
IBM watsonx.data can be deployed across multiple environments, including but not limited to IBM Cloud, Amazon Web Services (AWS) infrastructure, and on premises.

The storage layer is centered around object storage, which is highly available, highly scalable, and inexpensive.

A governance and metadata layer integrates existing Netezza and Db2 services to achieve metadata sharing using open-data formats such as Parquet, ORC, and Avro (a serialization format for record data and for streaming data pipelines) and leverages the Apache Iceberg table format. It uses multiple engines, such as Presto and Spark, which provide fast, reliable, efficient processing of big data at scale. Let us take a look at each layer in more detail.

Infrastructure

From an infrastructure layer perspective (Figure 2-7 on page 19), *quick start* steps enable organizations using Software as a Service (SaaS) tools to deploy in minutes, ready to bring and store their data into S3 object storage. Organizations may also choose to connect to existing data warehouses and look at the data using virtualization or federation techniques.



1 **Deploy seamlessly and fully managed in under 10 minutes**

- Data load ready in minutes
- Full service – storage, engine, and metastore all integrated and preconfigured

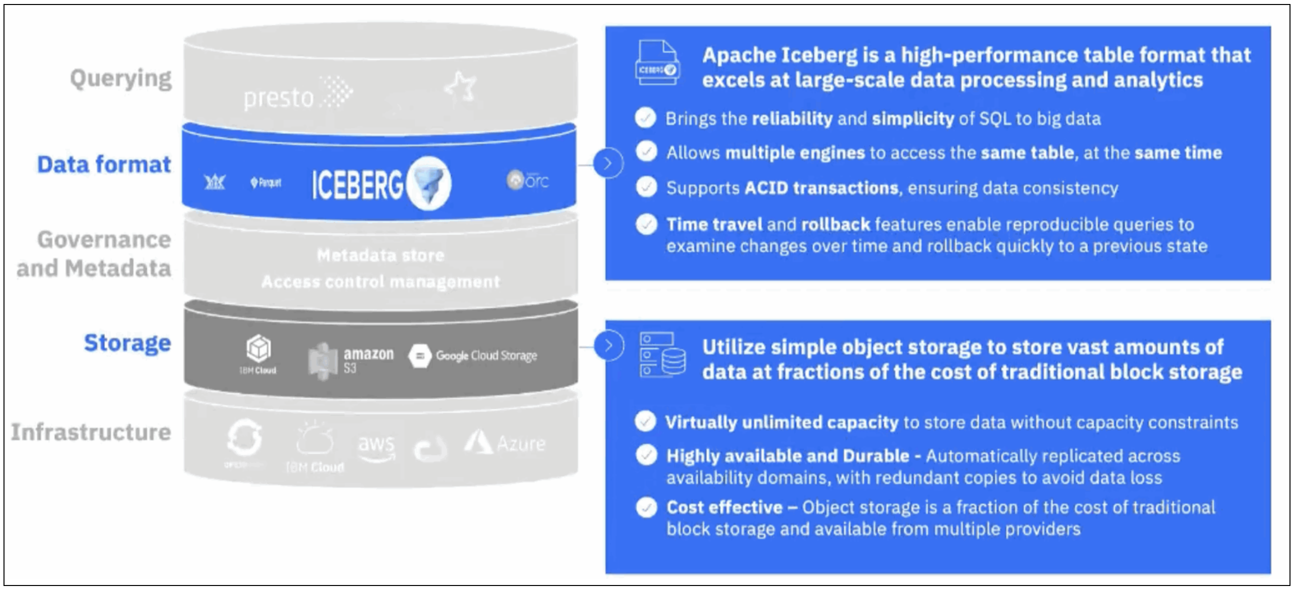
2 **Deploy across any cloud or on-premise environment**

- Combine on-premise and cloud data to create a unified view of data that is easier to manage, govern and analyze.
- Keep sensitive data on-premise while leveraging the scalability and cost-efficiency of the cloud

Figure 2-7 IBM watsonx.data infrastructure layer

Storage and data formats

IBM watsonx.data is designed to leverage various storage solutions, such as Amazon S3, IBM Cloud, Google Cloud Storage, and HDFS. Apache Iceberg helps solve the problem of bringing structure to data lakes. It is a metadata file that sits with the data files so that, as changes are made to the data, it keeps track of those records. It provides certain atomicity, consistency, isolation, and durability (ACID) transactional guarantees and the ability to roll back in time for any audit purposes. Organizations can view and understand the transactions that occurred and their completion status by looking back in time at previous states. See Figure 2-8.



Apache Iceberg is a high-performance table format that excels at large-scale data processing and analytics

- ✓ Brings the **reliability** and **simplicity** of SQL to big data
- ✓ Allows **multiple engines** to access the **same table**, at the **same time**
- ✓ Supports **ACID transactions**, ensuring data consistency
- ✓ **Time travel** and **rollback** features enable reproducible queries to examine changes over time and rollback quickly to a previous state

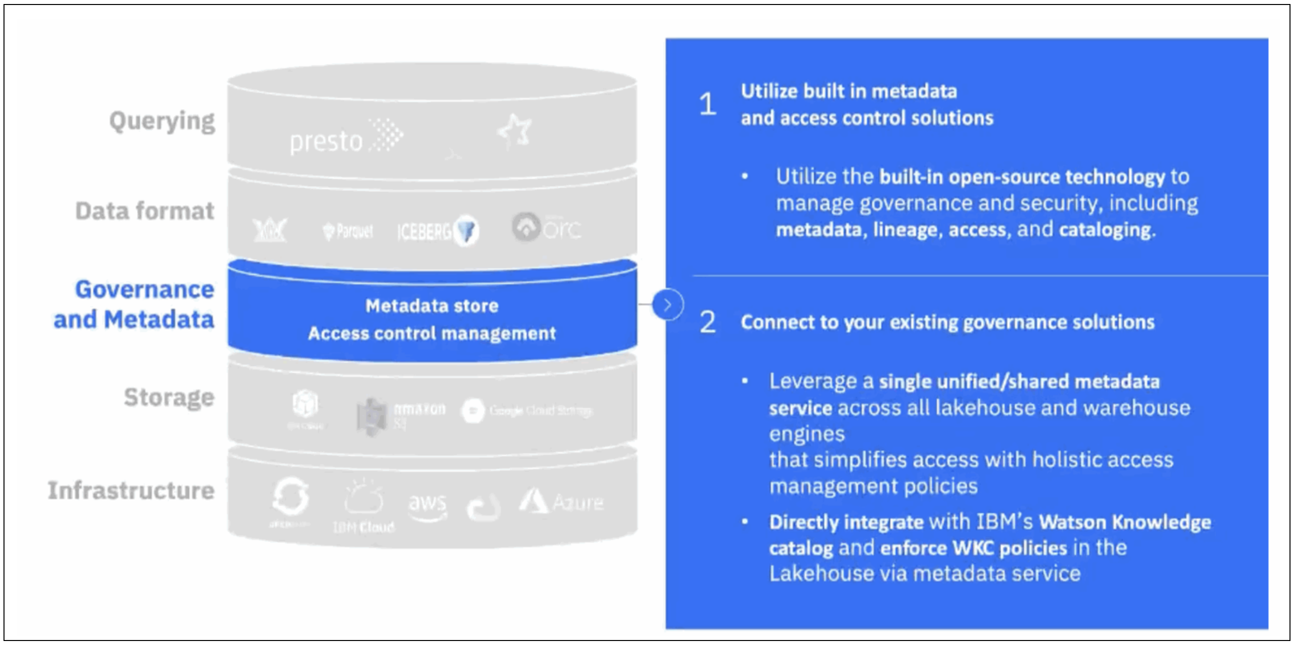
Utilize simple object storage to store vast amounts of data at fractions of the cost of traditional block storage

- ✓ **Virtually unlimited capacity** to store data without capacity constraints
- ✓ **Highly available and Durable -** Automatically replicated across availability domains, with redundant copies to avoid data loss
- ✓ **Cost effective –** Object storage is a fraction of the cost of traditional block storage and available from multiple providers

Figure 2-8 IBM watsonx.data storage and data format layers

Governance and metadata

The governance and metadata layer, shown in Figure 2-9, can be thought of as the glue that holds the multi-engine capabilities together. For example, it enables all engines to access the same storage, leverage the same table formats, and access the same metadata stores, thereby enabling organizations to look at the same sets of data through a unified catalog. Users are able to understand exactly what the data is, where it is, and what it looks like, no matter which query engine they are using. When a user changes or updates data in watsonx.data with an engine such as Presto, the metadata and catalog will be updated so that when that same user looks at that catalog later through a different query engine (for example, the Netezza engine), that user can continue exactly where they left off in watsonx.data.



1 **Utilize built in metadata and access control solutions**

- Utilize the **built-in open-source technology** to manage governance and security, including **metadata, lineage, access,** and **cataloging.**

2 **Connect to your existing governance solutions**

- Leverage a **single unified/shared metadata service** across all lakehouse and warehouse engines that simplifies access with holistic access management policies
- **Directly integrate** with IBM's **Watson Knowledge catalog** and **enforce WKC policies** in the Lakehouse via metadata service

Figure 2-9 IBM watsonx.data governance and metadata layer

Access-control management helps provide consistent governance across all watsonx.data lakehouse assets, integrating with IBM Knowledge Catalog capabilities to participate in global governance and providing a single source of the truth for policies and their enforcement. This is achieved through metadata integration and plug-ins to the engines.

There are three levels of user access controls in watsonx.data:

- **User authentication (Level 1 access control):** IBM watsonx.data works with a variety of Identity Provider Services, such as Identity Access Management (IAM) and Lightweight Directory Access Protocol (LDAP). Users who access the service through UIs, APIs, SQL editors, and command lines will be authenticated using their user ID and password, API keys, or authentication tokens.
- **User access to resources (Level 2 access control):** Roles can be assigned for watsonx.data users to access lakehouse resources. Roles at this level include Viewer, Editor, and Administrator. Resources include Instances, Engines, Catalogs. Storage, and Databases.
- **User access to data (Level 3 access control):** IBM watsonx.data enables data administrators to define data access policies for deeper levels of governance. Access policies can be defined on schemas, tables, columns, and rows.

Users are checked for access based on defined access policies. Advanced governance involving the masking of data, for example, leverages Watson Knowledge Catalog governance.

Querying

The querying layer, as previously mentioned, enables multiple query engines to coexist within watsonx.data. One size does not fit all when it comes to querying all the different types of data that may exist in a lakehouse. This layer enables the best engines to be intelligently assigned to query the target data sets for cost optimization of workloads. The different query engines are assigned infrastructure profiles. For example, Presto can leverage different flavors of worker nodes (used to run containerized applications and handle networking to ensure that traffic between applications across the cluster and from outside of the cluster can be properly facilitated).

One node might be used to better manage CPU-dense tasks that require heavy computing of arcane encryption. Another might be cache-optimized to cope with large data scans or work with large amounts of data that need to be close to the engine. By scaling out the worker nodes, organizations can be assured they have a sufficient cache to handle the workloads. All engines can be ephemeral and elastic as well as be usage-based so organizations can use instances of these engines to run their workloads, pause them, or delete them at will. Organizations are billed for only what they use and when they use it, scaling up and down to meet the necessary service-level agreements (SLAs). See Figure 2-10.

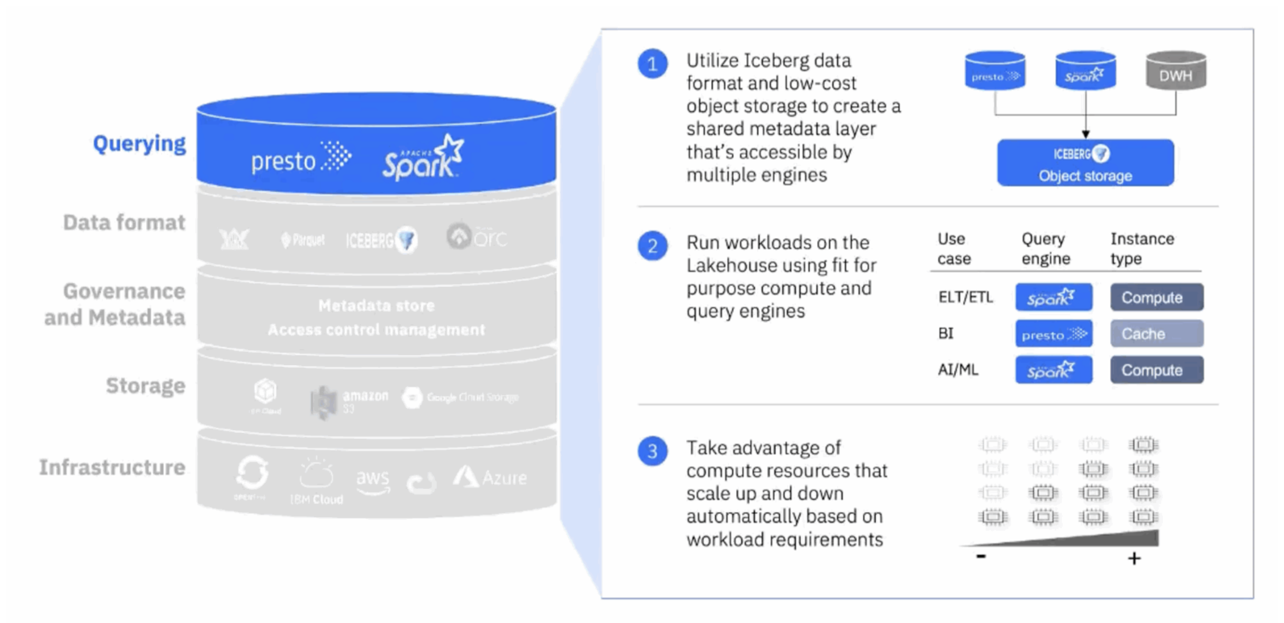

Querying
presto
Spark
Data format
Parquet
ICEBERG
orc
Governance and Metadata
Metadata store
Access control management
Storage
amazon S3
Google Cloud Storage
Infrastructure
aws
IBM Cloud
Azure
1
Utilize Iceberg data format and low-cost object storage to create a shared metadata layer that's accessible by multiple engines
presto
Spark
DWH
ICEBERG
Object storage
2
Run workloads on the Lakehouse using fit for purpose compute and query engines
Use case
Query engine
Instance type
ELT/ETL
Spark
Compute
BI
presto
Cache
AI/ML
Spark
Compute
3
Take advantage of compute resources that scale up and down automatically based on workload requirements
-
+

Figure 2-10 watsonx.data querying layer

2.3 Benefits of using watsonx.data for businesses

IBM watsonx.data is designed to help organizations:

- Access all their data and maximize workload coverage across all hybrid-cloud environments. Expect seamless deployment of a fully managed service across any cloud or on-premises environment. Access any data source, wherever it resides, through a single point of entry and combine it using open data formats.

- Integrate into existing environments with open source, open standards, and interoperability with IBM and third-party services.
- Accelerate time to trusted insights. Start with built-in governance and automation; strengthen enterprise compliance and security with unified governance across the entire ecosystem. A click-and-go console helps teams ingest, access, and transform data and run workloads. The product provides a dashboard that makes it easier for organizations to save money and deliver fresh, trusted insights.
- Reduce the cost of a data warehouse by up to 50%[1] through workload optimization across multiple query engines and storage tiers. Optimize costly warehouse workloads with fit-for-purpose engines that scale up and down automatically. Reduce costs by eliminating duplication of data when the enterprise uses low-cost object storage; extract more value from the data in ineffective data lakes.

Some of the key capabilities of watsonx.data are:

- Scaling for BI across all data with multiple high-performance query engines optimized for different workloads (for example: Presto, Spark, Db2, Netezza, and so forth.
- Enabling data-sharing between these different engines.
- Using shared common data storage across data lake and data warehouse functions, avoiding unnecessary time-consuming ETL/ELT jobs.
- Eradicating unnecessary data duplication and replication.
- Providing consistent governance, security, and user experience across hybrid multi-clouds.
- Leveraging an open and flexible architecture built on open source without vendor lock-in.
- The ability to be deployed across hybrid-cloud environments (on-premises, private, public clouds) on multiple hyperscalers.
- Offering a wide range of prebuilt integration capabilities incorporating IBM data-fabric capabilities.
- Providing global governance across all data in the enterprise, leveraging the IBM data-fabric capabilities.
- Extensibility through APIs, value-add partner ecosystems, accelerators, and third-party solutions.

2.4 Data pipeline considerations for open lakehouse

When designing a data pipeline for an open lakehouse architecture, several considerations must be addressed to ensure efficient, scalable, and robust data handling. The integration of tools like Apache Spark, ETL tools like DataStage®, and StreamSets plays a crucial role in shaping the pipeline's functionality.

2.4.1 Apache Spark: The computational engine

Apache Spark serves as a distributed data processing engine, capable of handling large-scale data with its in-memory computation capabilities. In an open lakehouse, Spark is often the go-to engine for data transformation, analytics, and machine learning workloads.

[1] This cost reduction was calculated by comparing published 2023 list prices normalized for virtual private cloud (VPC) hours of IBM watsonx.data to several major cloud data warehouse vendors. Savings may vary depending on configurations, workloads, and vendors.

The following are the key considerations:

- **Scalability**: Spark's distributed architecture enables horizontal scaling, handling vast datasets stored in the lakehouse.
- **Batch and streaming**: Supports both batch and real-time data processing, crucial for a unified lakehouse approach.
- **Interoperability**: Works seamlessly with various data formats (Parquet, Delta Lake, ORC, and so forth) typically used in open lakehouses.
- **Optimization**: Employ Spark SQL and Catalyst optimizer for query performance tuning.
- **Cost efficiency**: Leverage the decoupled storage and compute paradigm of lakehouses, optimizing resource utilization.

2.4.2 ETL tools: Managing complex workflows

IBM DataStage provides robust Extract, Transform, and Load (ETL) capabilities that are crucial for cleaning, transforming, and loading structured and semi-structured data into the lakehouse.

The following are key considerations:

- **Data quality and governance**: Ensures data is accurate, consistent, and compliant with governance policies before ingestion.
- **Workflow orchestration**: Simplifies complex ETL workflows and supports dependency management between tasks.
- **Error handling**: Offers robust logging and error-handling mechanisms to ensure data pipeline reliability.
- **Integration**: Interfaces well with data lakes, data warehouses, and third-party systems for seamless data flow.
- **Reusability**: Reusable jobs and transformations improve development speed and maintainability.

2.4.3 StreamSets: Real-time data integration and monitoring

StreamSets is a powerful platform for real-time data integration and monitoring. It facilitates continuous data ingestion from multiple sources into the lakehouse.

The following are the key considerations:

- **Streaming pipelines**: Supports real-time ingestion with minimal latency, critical for time-sensitive data.
- **Data drift management**: Automatically adapts to schema changes, reducing pipeline maintenance overhead.
- **Observability**: Provides end-to-end visibility and monitoring of data pipelines, enabling proactive issue resolution.
- **Data enrichment**: Allows data enrichment and transformation in motion, reducing load on downstream systems.
- **Integration**: Can connect with a variety of streaming sources (Kafka, IoT devices) and targets (Delta Lake, cloud storage).

2.4.4 General design considerations for open lakehouse pipelines

The following are key design considerations for open lakehouse pipelines:

- **Data formats**: Use open, columnar formats like Parquet or Delta for storage efficiency and analytics performance.
- **Schema evolution**: Ensure support for schema changes over time without breaking the pipeline.
- **Data governance**: Implement robust governance for security, access control, and data lineage.
- **Performance tuning**: Optimize resource allocation, caching strategies, and data partitioning.
- **Fault tolerance**: Design for resilience with retry mechanisms, checkpointing (especially for streaming), and redundancy.

The combination of Apache Spark, ETL tools like DataStage, and StreamSets creates a powerful, flexible, and efficient pipeline for open lakehouse architectures. Spark handles intensive computation and analytics, DataStage ensures structured ETL processes, and StreamSets provides real-time ingestion and monitoring. Together, these tools facilitate seamless data flow, governance, and scalability across the pipeline.

2.5 Data pipelines integration with watsonx.data

The integration of tools like Apache Spark, IBM DataStage, and StreamSets with watsonx.data becomes essential for building robust, scalable, and efficient data pipelines. watsonx.data is designed to unify data across various formats and environments, enabling advanced analytics and AI workloads. Here is how these tools fit into the pipeline:

2.5.1 Apache Spark in watsonx.data

Apache Spark serves as the primary engine for distributed data processing within watsonx.data. Its integration is key to executing high-performance, large-scale data transformations and analytics.

The following are the key considerations:

- **Unified analytics**: Spark, within watsonx.data, processes data directly in the lakehouse, supporting SQL queries, ML, and graph analytics.
- **Delta lake integration**: Enables ACID transactions, schema enforcement, and time travel, ensuring reliable and consistent data for analytics.
- **Batch and streaming workloads**: Spark supports both batch processing (for example, historical data analysis) and streaming (real-time data ingestion) using Structured Streaming.
- **Optimization**: Use the watsonx.data built-in query accelerators and Spark’s Catalyst optimizer for efficient query execution and resource usage.
- **Interoperability**: Spark seamlessly integrates with various data formats supported by watsonx.data (for example, Parquet, Delta Lake).

2.5.2 IBM DataStage in watsonx.data

IBM DataStage is essential for ETL operations, focusing on transforming and loading data into watsonx.data for downstream consumption.

The following are the key considerations:

- **Data integration**: DataStage ensures smooth data ingestion from on-premises, cloud, and third-party systems into watsonx.data.
- **Data governance and quality**: Provides tools for data cleansing, transformation, and validation before ingestion, critical for maintaining data quality in the lakehouse.
- **Workload automation**: Automates ETL jobs to handle large volumes of structured and semi-structured data efficiently.
- **Orchestration and scheduling**: Manages complex workflows and schedules data processing tasks within watsonx.data.
- **Data lineage**: Ensures full traceability of data transformations, aligning with governance policies in watsonx.data.

2.5.3 StreamSets in watsonx.data

StreamSets complements watsonx.data by enabling real-time data ingestion and continuous data integration from various streaming sources.

The following are the key considerations:

- **Real-time ingestion**: Facilitates streaming data into watsonx.data for near real-time analytics, critical for use cases like IoT, log analysis, and fraud detection.
- **Data drift management**: Automatically adapts to schema changes, reducing the need for manual intervention in dynamic environments.
- **Observability and monitoring**: Provides end-to-end visibility into data pipelines, helping identify bottlenecks and ensuring data reliability.
- **Integration with hybrid sources**: Supports seamless connectivity to Kafka, cloud storage, and on-premise systems, aligning with the watsonx.data hybrid cloud strategy.
- **Data transformation in motion**: Allows real-time enrichment and transformation of data before it lands in watsonx.data.

2.5.4 IBM watsonx.data benefits from this ecosystem

IBM watsonx.data integrates seamlessly with existing tools like Apache Spark, DataStage, and StreamSets. In this section we discuss the benefits from this ecosystem.

Data unification and accessibility

The following benefits for data unification and accessibility can be achieved:

- watsonx.data leverages open formats (for example, Parquet, Delta Lake) to store and process data, making it accessible to all three tools (Spark, DataStage, StreamSets).
- It provides a single source of truth for AI and analytics workloads.

Scalability and performance

There are several scalability and performance benefits, such as:

- Apache Spark ensures scalable compute for large-scale analytics.

- DataStage efficiently manages ETL workflows to prepare high-quality data.
- StreamSets delivers real-time data with low latency, keeping watsonx.data pipelines dynamic and up-to-date.

Governance and compliance

Key governance and compliance features include:

- Built-in governance features in watsonx.data, enhanced by DataStage's data quality and lineage capabilities, ensure compliance with regulatory requirements.
- StreamSet's monitoring capabilities add transparency, aligning with the watsonx.data data governance framework.

Hybrid and open ecosystem

watsonx.data's open and hybrid design allows integration across cloud and on-premises environments, fully leveraging the strengths of Spark, DataStage, and StreamSets.

In a watsonx.data lakehouse architecture, the combined power of Apache Spark, IBM DataStage, and StreamSets ensures a robust data pipeline. Spark handles complex analytics and ML workloads, DataStage ensures ETL processes are efficient and governed, while StreamSets brings real-time capabilities. Together, they enable organizations to harness the full potential of watsonx.data for data-driven decision-making and AI innovation.

2.5.5 Synergy between watsonx.data and other watsonx platform components

IBM watsonx platform is designed to work cohesively, with each component strengthening the others. Here is how watsonx.data specifically synergizes with other components to improve the overall AI experience

Synergy between watsonx.data and watsonx.ai

IBM watsonx.data and watsonx.ai work together to provide a seamless pipeline for data-driven AI initiatives. watsonx.data serves as a high-performance, scalable data lakehouse designed to store and process both structured and unstructured data. It optimizes data access for AI workloads by integrating tools for querying, data transformation, and analysis. This foundation enables watsonx.ai: a platform focused on building, deploying, and managing AI models, to leverage high-quality, well-prepared data for training and inferencing. Together, watsonx.data ensures that AI workflows powered by watsonx.ai are fueled by reliable and optimized data sources, reducing preprocessing bottlenecks and accelerating insights. By integrating these solutions, organizations can streamline the development of advanced AI applications with precision and efficiency, ensuring a robust end-to-end AI lifecycle.

Synergy between watsonx.data and watsonx.governance

The synergy between watsonx.data and watsonx.governance creates a framework of compliance, trust, and accountability for data management and AI deployment. watsonx.data provides the infrastructure for managing the vast amounts of data necessary for AI development. watsonx.governance overlays this with a layer that enforces ethical AI principles (for example, fairness), monitors compliance, and manages risk. This combination helps assure data utilized in AI models adheres to organizational and regulatory standards, mitigating risks of bias or misuse. By connecting these tools, enterprises can scale their AI initiatives confidently, knowing that data and model governance are baked into their workflows. This integration fosters transparency and accountability, which are critical for building trust in AI systems.

These synergies provide the backbone for scalable, trustworthy, and efficient AI operations, ensuring that businesses can innovate while maintaining compliance and reliability.

3

Ingesting data into an open data lakehouse

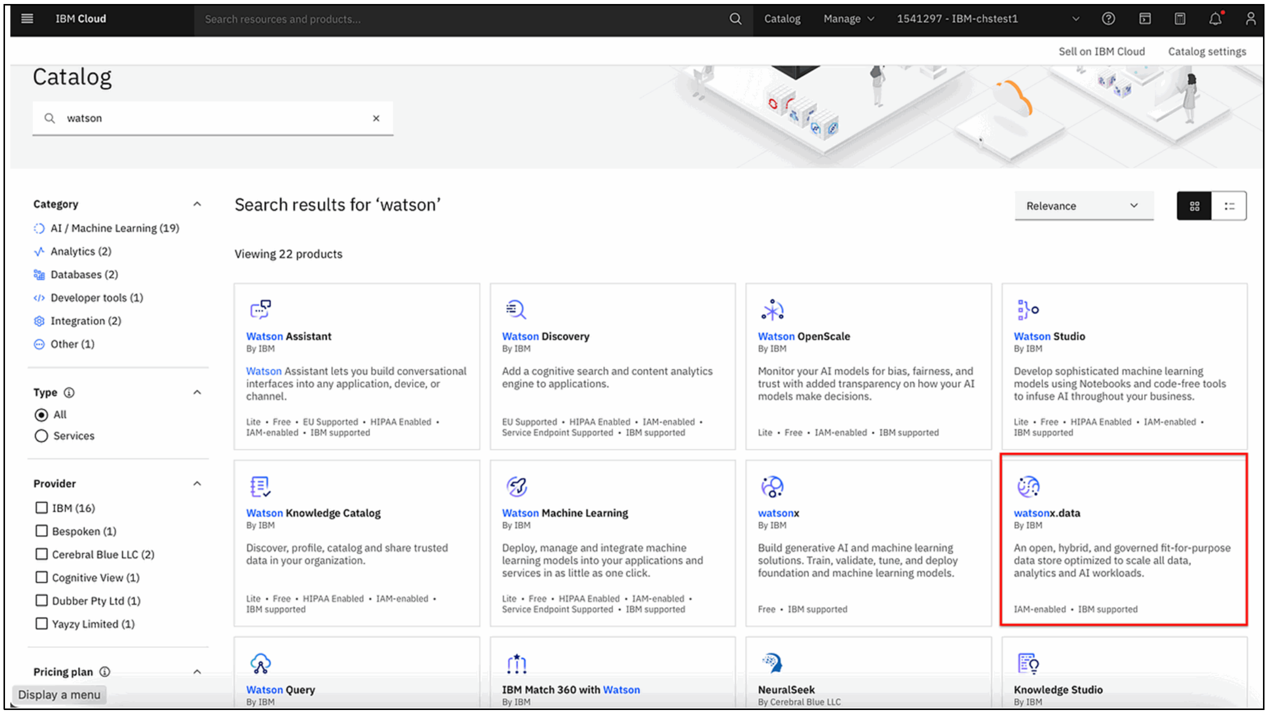
The foundation of any data-driven organization is its ability to efficiently ingest data from diverse sources. IBM watsonx.data, which is a powerful open data lakehouse platform, streamlines this process by providing a scalable and secure environment for data ingestion. This approach simplifies data intake so that you can unlock the full potential of your data for advanced analytics and AI-powered insights.

This chapter covers how to ingest data into an open data lakehouse in watsonx.data and has the following sections:

- “Provisioning and configuring IBM watsonx.data” on page 30
- “Integrating external data sources: Federation in PrestoDB” on page 30
- “Techniques for ingesting data (structured or unstructured)” on page 35
- “Ingesting data” on page 36
- “Data pipeline considerations for open lakehouse” on page 41

3.1 Provisioning and configuring IBM watsonx.data

To explore IBM watsonx.data on IBM Cloud, go to the `cloud.ibm.com` catalog and search for “watsonx.data”. Use the default options for a streamlined initial experience, as shown in Figure 3-1.



Search results for ‘watson’

Viewing 22 products

Watson Assistant
By IBM
Watson Assistant lets you build conversational interfaces into any application, device, or channel.
Lite • Free • EU Supported • HIPAA Enabled • IAM-enabled • IBM supported

Watson Discovery
By IBM
Add a cognitive search and content analytics engine to applications.
EU Supported • HIPAA Enabled • IAM-enabled • Service Endpoint Supported • IBM supported

Watson OpenScale
By IBM
Monitor your AI models for bias, fairness, and trust with added transparency on how your AI models make decisions.
Lite • Free • IAM-enabled • IBM supported

Watson Studio
By IBM
Develop sophisticated machine learning models using Notebooks and code-free tools to infuse AI throughout your business.
Lite • Free • HIPAA Enabled • IAM-enabled • IBM supported

Watson Knowledge Catalog
By IBM
Discover, profile, catalog and share trusted data in your organization.
Lite • Free • HIPAA Enabled • IAM-enabled • IBM supported

Watson Machine Learning
By IBM
Deploy, manage and integrate machine learning models into your applications and services in as little as one click.
Lite • Free • HIPAA Enabled • IAM-enabled • Service Endpoint Supported • IBM supported

watsonx
By IBM
Build generative AI and machine learning solutions. Train, validate, tune, and deploy foundation and machine learning models.
Free • IBM supported

watsonx.data
By IBM
An open, hybrid, and governed fit-for-purpose data store optimized to scale all data, analytics and AI workloads.
IAM-enabled • IBM supported

Watson Query
By IBM

IBM Match 360 with Watson
By IBM

NeuralSeek
By Cerebral Blue LLC

Knowledge Studio
By IBM

Figure 3-1   IBM watsonx.data on IBM Cloud

3.2 Integrating external data sources: Federation in PrestoDB

Several database systems are accessible both within and outside a virtual machine environment, such as IBM watsonx.data Presto, Db2, MySQL, and PostgreSQL. To access these databases externally, you need the server name, port number for the service, and the `presto-key.jks` file for connecting to Presto.

3.2.1 Connecting to IBM watsonx.data Presto

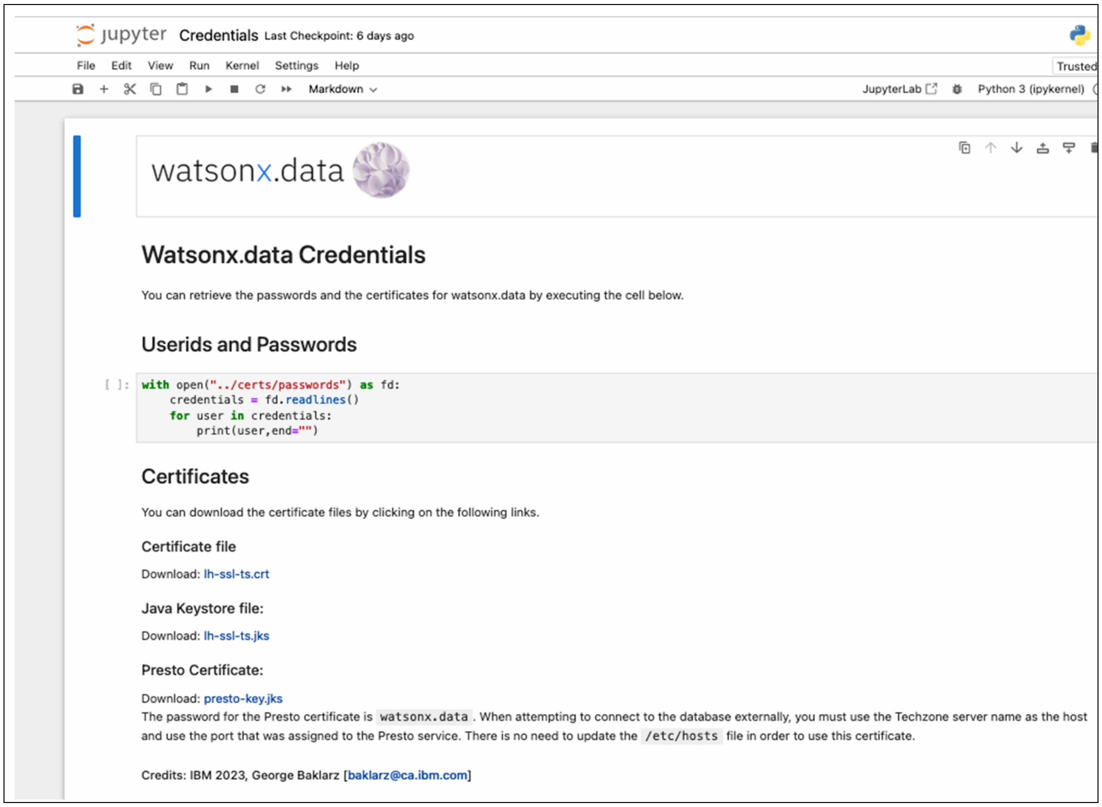
When connecting to the IBM watsonx.data Presto database, a connection certificate must be available on the client machine (typically your workstation) or another service like IBM Cloud Pak for Data. To obtain the certificate, complete the following steps:

1. To extract the certificate to your local file system, use the following command in a terminal window. Replace the port and `region.services.cloud.techzone.com` with the SSH values that are found in the TechZone reservation.

   ```
   scp -P port
   watsonx@region.services.cloud.techzone.ibm.com:/certs/presto-key.jks
   /Users/myname/Downloads
   ```

   (You need to change this local path to your own path).

2. Download the certificate from the Jupyter Notebook's Credentials notebook, as shown in Figure 3-2.


jupyter Credentials Last Checkpoint: 6 days ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)
watsonx.data

Watsonx.data Credentials

You can retrieve the passwords and the certificates for watsonx.data by executing the cell below.

Userids and Passwords

```
[ ]: with open("../certs/passwords") as fd:
         credentials = fd.readlines()
         for user in credentials:
             print(user,end="")
```

Certificates

You can download the certificate files by clicking on the following links.

Certificate file

Download: lh-ssl-ts.crt

Java Keystore file:

Download: lh-ssl-ts.jks

Presto Certificate:

Download: presto-key.jks
The password for the Presto certificate is `watsonx.data`. When attempting to connect to the database externally, you must use the Techzone server name as the host and use the port that was assigned to the Presto service. There is no need to update the `/etc/hosts` file in order to use this certificate.

Credits: IBM 2023, George Baklarz [baklarz@ca.ibm.com]

Figure 3-2 Downloading the certificate from the Jupyter Notebook's Credentials notebook

3. When connecting to the Presto engine, choose the PrestoDB driver. For local access, the following credentials are used:

   – Hostname: `localhost`
   – Port: `8443`
   – Username: `ibmlhadmin`
   – Password: `password`
   – Database: `tpch`
   – SSL: `True`
   – SSLTrustStorePath: `/certs/presto-key.jks`
   – SSLTrustStorePassword: `watsonx.data`

> **Note:** The IBM watsonx.data Presto database prioritizes secure communication. To achieve this goal, it relies on a client-side certificate for authentication. This certificate acts as a digital key that verifies your identity when connecting to the database. When using a Jupyter Notebook environment with IBM watsonx.data Presto, the certificate is available within a dedicated notebook that is named Credentials, as shown in Figure 3-2. This notebook contains links for downloading the necessary certificate files.

4. In the following settings, replace the Hostname and Port placeholders with the values from your TechZone reservation.

   Here are the database connection settings

   - Hostname: `region.services.cloud.techzone.ibm.com`
   - Port: `port`
   - Username: `ibmlhadmin`
   - Password: `password`
   - Database: `tpch`

   Set the following driver properties:

   - `SSL True`
   - `SSLTrustStorePath /mydownload/presto-key.jks`
   - `SSLTrustStorePassword watsonx.data`

5. The `/mydownload/presto-key.jks` value must be replaced with the location that you copied the key from in step 1 on page 30. When connecting to the Db2 engine, select the Db2 LUW driver. The Db2 server can be accessed on port 50000 inside the virtual machine by using the following credentials:

   - Hostname: `watsonxdata`
   - Port: `50000`
   - Username: `db2inst1`
   - Password: `db2inst1`
   - Database: `gosales`
   - SSL: `off`

6. When accessing the database outside the virtual machine, you must change the host to `region.services.cloud.techzone.ibm.com` and the `port number` based on your TechZone reservation. All the other settings remain the same.

   - Hostname: `region.services.cloud.techzone.ibm.com`
   - Port: `port`
   - Username: `db2inst1`
   - Password: `db2inst1`
   - Database: `gosales`
   - SSL: `off`

3.2.2 PostgreSQL access

When connecting to the PostgreSQL engine, select the PostgreSQL driver. To connect to the PostgreSQL system, extract the admin password by using the **`cat /certs/passwords`** command when connected to the IBM watsonx.data system.

You can also retrieve the credentials by opening the Credentials notebook in the Jupyter Notebook service. When accessing the PostgreSQL database in the system, use the following settings.

- Hostname: `ibm-lh-postgres`
- Port: `5432`
- Username: `admin`
- Password: The value that was extracted with the **`cat /certs/passwords`** command in the previous step
- Database: *gosalesdw*

3.2.3 PostgreSQL external access

The following credentials are used for remote access.

- Hostname: `region.services.cloud.techzone.com`
- Port: `port`
- Username: `admin`
- Password: The value that was extracted that was extracted with the **`cat /certs/passwords`** command in the previous step
- Database name: `gosalesdw`

3.2.4 MySQL access

When connecting to the MySQL engine, select the MySQL driver by running the following command:

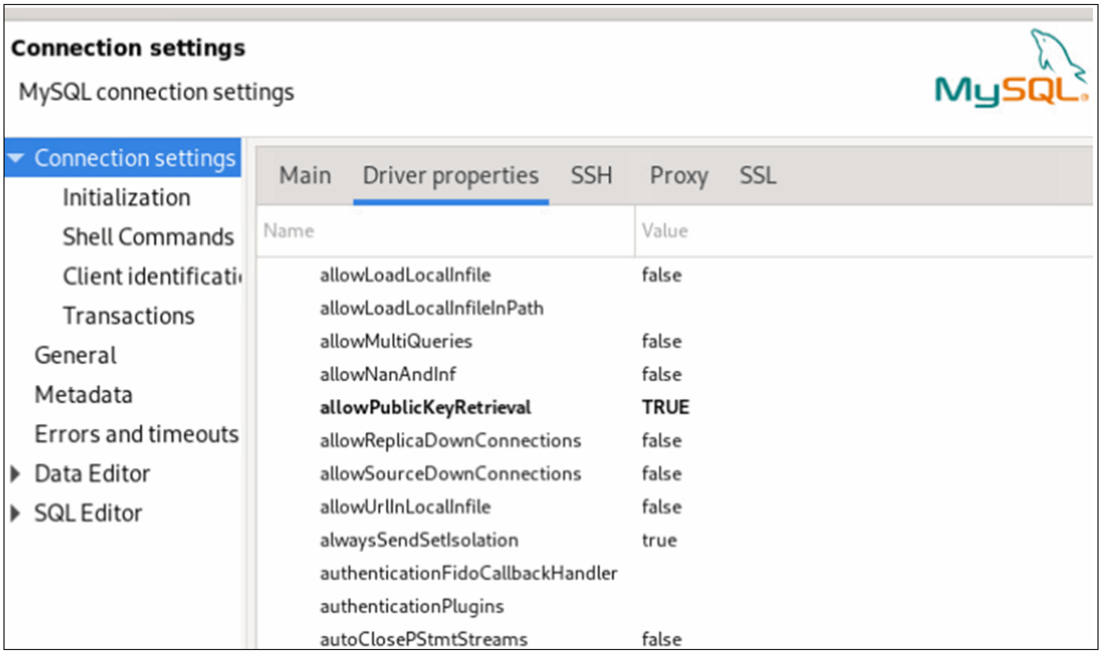
```
export POSTGRES_PASSWORD=$(docker exec ibm-lh-postgres printenv | grep
POSTGRES_PASSWORD | sed 's/.*=//')
echo "Postgres Userid   : admin"
echo "Postgres Password : " $POSTGRES_PASSWORD
echo $POSTGRES_PASSWORD > /tmp/postgres.pw
```

3.2.5 MySQL internal access

When accessing the MySQL database in the system, use the following settings:

- Hostname: `watsonxdata`
- Port: `3306`
- Username: `root`
- Password: `password`
- Database: `gosalesdw`

Set `allowPublicKeyRetrieval` to `True` for the connection to work with dBeaver, as shown in Figure 3-3.



Connection settings

MySQL connection settings

Name	Value
allowLoadLocalInfile	false
allowLoadLocalInfileInPath	
allowMultiQueries	false
allowNanAndInf	false
allowPublicKeyRetrieval	**TRUE**
allowReplicaDownConnections	false
allowSourceDownConnections	false
allowUrlInLocalInfile	false
alwaysSendSetIsolation	true
authenticationFidoCallbackHandler	
authenticationPlugins	
autoClosePStmtStreams	false

Figure 3-3 MySQL internal access

3.2.6 MySQL external access

The following credentials are used for remote access.

- Hostname: `region.services.cloud.techzone.com`
- Port: `port`
- Username: `root`
- Password: `password`
- Database name: `gosalesdw`

Set `allowPublicKeyRetrieval` to `True` for the connection to work with dBeaver, as shown in Figure 3-3.

After adding a database, wait a few moments before attempting access. The Presto server requires a brief startup period. To verify its readiness, run the `check_presto` command in a terminal window. Wait until it confirms that the service is `ready`.

When adding database engines to your IBM watsonx.data system, ensure that each has a unique display name. Although you might initially use the same name as the original database (for example, `gosales` for both Db2 and PostgreSQL), this approach can lead to conflicts later. For example, if you add the PostgreSQL database to the system, the display name cannot be the same.

> **Tip:** You might want to differentiate databases with the same name by prefixing them with the database type. For example, `db2_gosales` for Db2 and `pg_gosales` for PostgreSQL.

It might take a few minutes for the database contents to appear. Refresh the browser window if no changes are visible after this time.

3.3 Techniques for ingesting data (structured or unstructured)

Data ingestion, the process of bringing your data into IBM watsonx.data, is simple and secure for the following reasons:

- Visual interface: The **Ingest data** tab in the Data manager page offers an interface for loading data into IBM watsonx.data.
- Flexible data sources: Choose between ingesting local files or remote data sources to create tables by using the **Create table from file** option.
- Automatic schema inference: You do not need to predefine the table schema because IBM watsonx.data intelligently discovers the structure of your data (schema) during the first query execution.
- Consistent data format: Help ensure that your data files are in the same format type with a consistent structure (schema) for a smooth ingestion process.

IBM watsonx.data auto-discovers the schema based on the source file being ingested:

- Presto is used as the engine for running your SQL-based queries. Presto is an open-source SQL query engine that can work with data in several different data sources. You can also use it to load and ingest huge amounts of data in IBM watsonx.data.
- Apache Iceberg is a robust table format that is designed for managing large, evolving data sets. Unlike traditional formats like Parquet or CSV, Iceberg offers advanced features such as data snapshots, schema evolution, and data compaction. These capabilities make it a powerful tool for maintaining and optimizing your data lakehouse. Also, Iceberg introduces ACID transactions to the data lakehouse, a previously unavailable feature that you can use to perform atomic updates and deletions on tables, which helps ensure data consistency and integrity.
- Hive Meta Store serves as a central repository for storing meta-data about Apache Hive tables. This meta-data includes the table schema, data locations, and other crucial information that is necessary for managing and querying data within the Hive ecosystem.

  For seamless data management, IBM watsonx.data uses the Hive Metastore as its back-end system, which enables the platform to maintain, manage, and catalog the meta-data that is associated with tables, like Hive itself. This integration helps ensure efficient data discovery and simplifies querying within IBM watsonx.data.
- Apache Spark is a powerful open-source, distributed processing system that is designed to handle big data workloads efficiently. Spark's efficient data processing capabilities make it a powerful tool for executing critical Iceberg table operations like updates, deletes, and merges, ensuring data integrity and consistency in data warehouses. Spark offers even more value beyond this specific use case:
  - Data processing: Spark excels at data processing tasks like filtering and transforming raw data. Its procedural code capabilities enable efficient manipulation compared to pure SQL. You can process data with Spark and then store it in an S3-compatible object storage for later querying with Presto.
  - Advanced analytics: Data in IBM watsonx.data, such as data from Db2 or other ingested sources, might be better suited for analysis by using Spark's procedural code rather than SQL. Spark's capabilities can unlock deeper insights.
  - Flexible data ingestion: Spark provides a wide range of options for data ingestion, which include streaming data, data frames, Resilient Distributed Datasets (RDDs), and support for various data formats. This versatility simplifies integrating data from diverse sources.

Each component in the IBM watsonx.data stack provides unique capabilities. In IBM watsonx.data, these different components work together in tandem to provide the best data management capability for your analytics workloads.

Here are some of the key requirements of the Ingestion tool:

- The target table must be an Iceberg format table.
- IBM Storage Ceph, IBM Cloud Object Storage, AWS S3, and MinIO object storage are supported.
- Parquet, CSV, JSON, ORC, and AVRO file formats are supported as source data files.
- The maximum limit for the cumulative size of files must be within 500 MB for local ingestion.
- Parquet files exceeding 2 MB cannot be previewed, but they are ingested successfully.
- JSON files with complex nested objects and arrays are not previewed in the UI.
- Complex JSON files are ingested as arrays, which might hinder optimal data visualization and analysis.
- JSON files exceeding 2 MB cannot be previewed, but they are ingested successfully.
- Keys within JSON files must be enclosed in quotation marks for proper parsing and interpretation.
- AVRO files exceeding 2 MB cannot be previewed, but they are ingested successfully.
- ORC files exceeding 2 MB cannot be previewed, but they are ingested successfully.

3.4 Ingesting data

To ingest data into IBM watsonx.data, you can use the intuitive user interface or the command-line interface (CLI) that is provided by the `ibm-lh` tool. With the user interface, you can easily upload data files, define schemas, and transform data as needed. Alternatively, the **`ibm-lh`** tool offers granular control over the ingestion process so that you can automate data pipelines and integrate with existing workflows. By suing these methods, you can seamlessly bring your data into IBM watsonx.data, where it can be transformed, analyzed, and used to drive valuable insights.

3.4.1 Loading or ingesting data through the CLI

IBM watsonx.data uses the `ibm-lh` tool for managing ingestion jobs. To initiate an ingestion job, install `ibm-lh-client` locally. This client provides the CL for interacting with the `ibm-lh` tool and triggering ingestion processes. For more information and instructions about installing the `ibm-lh-client` package and using the `ibm-lh` tool for ingestion, see Installing ibm-lh-client and Setting up the ibm-lh command-line utility.

The `ibm-lh` tool supports the following features:

- Auto-discovery of schema based on the source file or target table.
- Advanced table configuration options for the CSV files:
  - Delimiter
  - Header
  - File encoding
  - Line delimiter
  - Escape characters

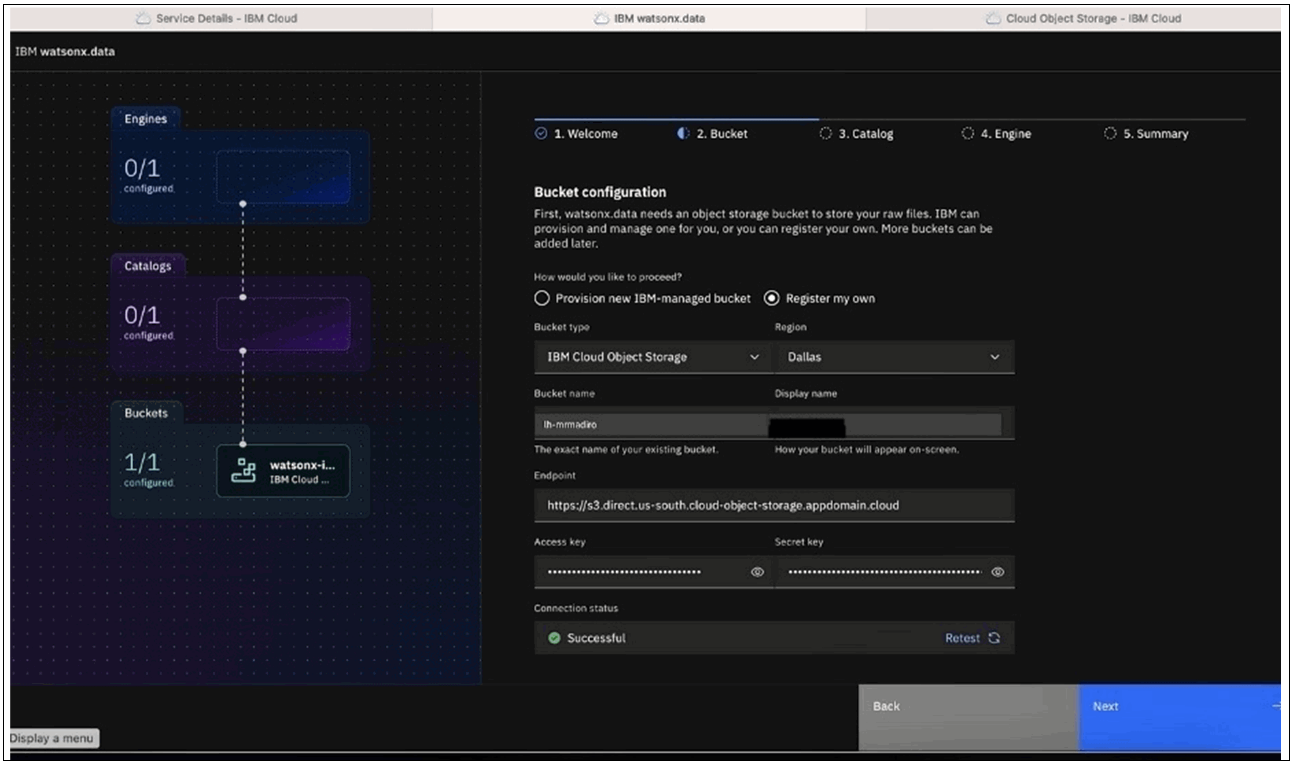
- Ingestion of single, multiple files, or a single folder (no subfolders) of S3 and local Parquet files.
- Ingestion of single, multiple files, or a single folder (no subfolders) of S3 and local CSV files.

3.4.2 Configuring an S3 IBM Cloud Object Storage bucket

IBM watsonx.data uses an object storage bucket to store data and its associated metadata. During setup, you can choose between two options:

- Automatic Bucket Creation: IBM watsonx.data creates a bucket for you.
- Pre-Existing Bucket: You can specify a bucket that you already created.

Figure 3-4 illustrates using a pre-existing bucket that is named `lh-xxx`.



Bucket configuration

First, watsonx.data needs an object storage bucket to store your raw files. IBM can provision and manage one for you, or you can register your own. More buckets can be added later.

How would you like to proceed?

○ Provision new IBM-managed bucket ● Register my own

Bucket type: IBM Cloud Object Storage

Region: Dallas

Bucket name: lh-mmadiro

The exact name of your existing bucket.

Display name: [redacted]

How your bucket will appear on-screen.

Endpoint: https://s3.direct.us-south.cloud-object-storage.appdomain.cloud

Access key: ••••••••••••••••••••••••••••••••

Secret key: ••

Connection status: Successful

Figure 3-4 Configuring a S3 or an IBM Cloud Object Storage bucket

3.4.3 Choosing the catalog

For this scenario, select **Apache Iceberg**, as shown in Figure 3-5.

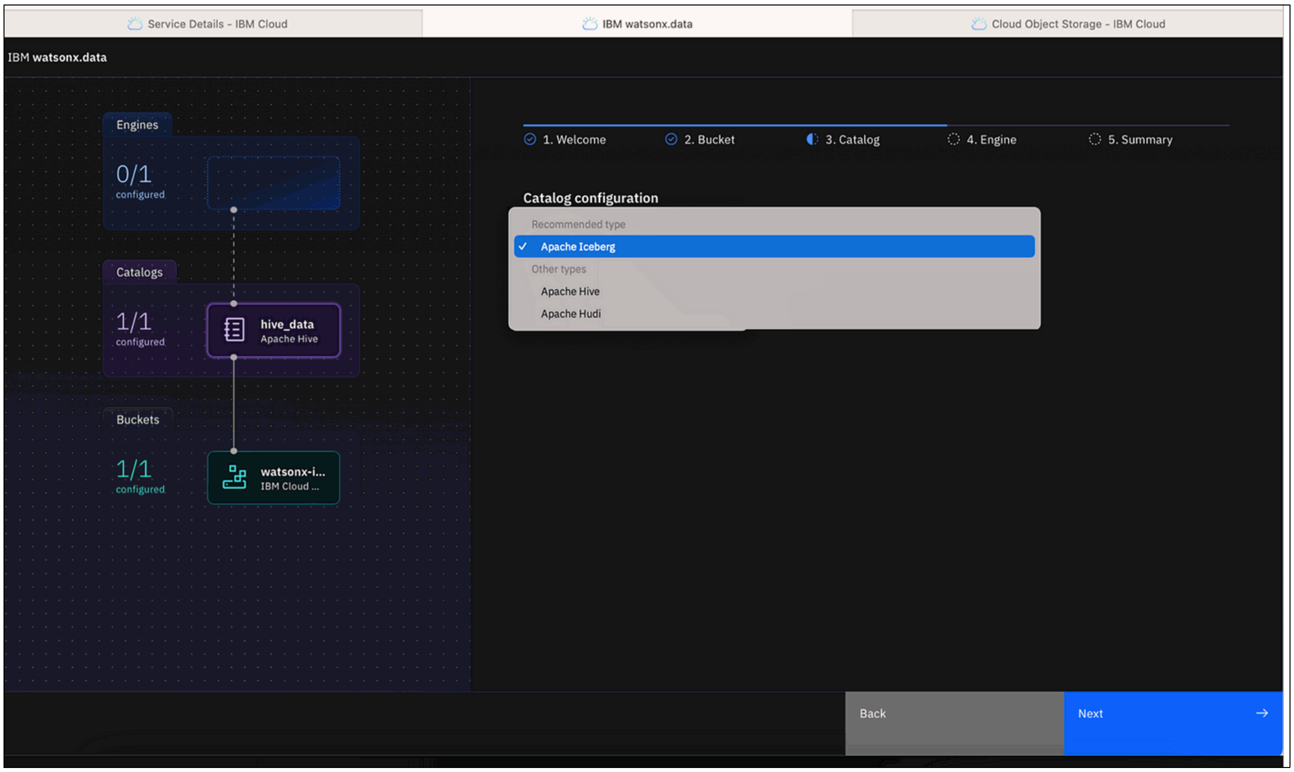
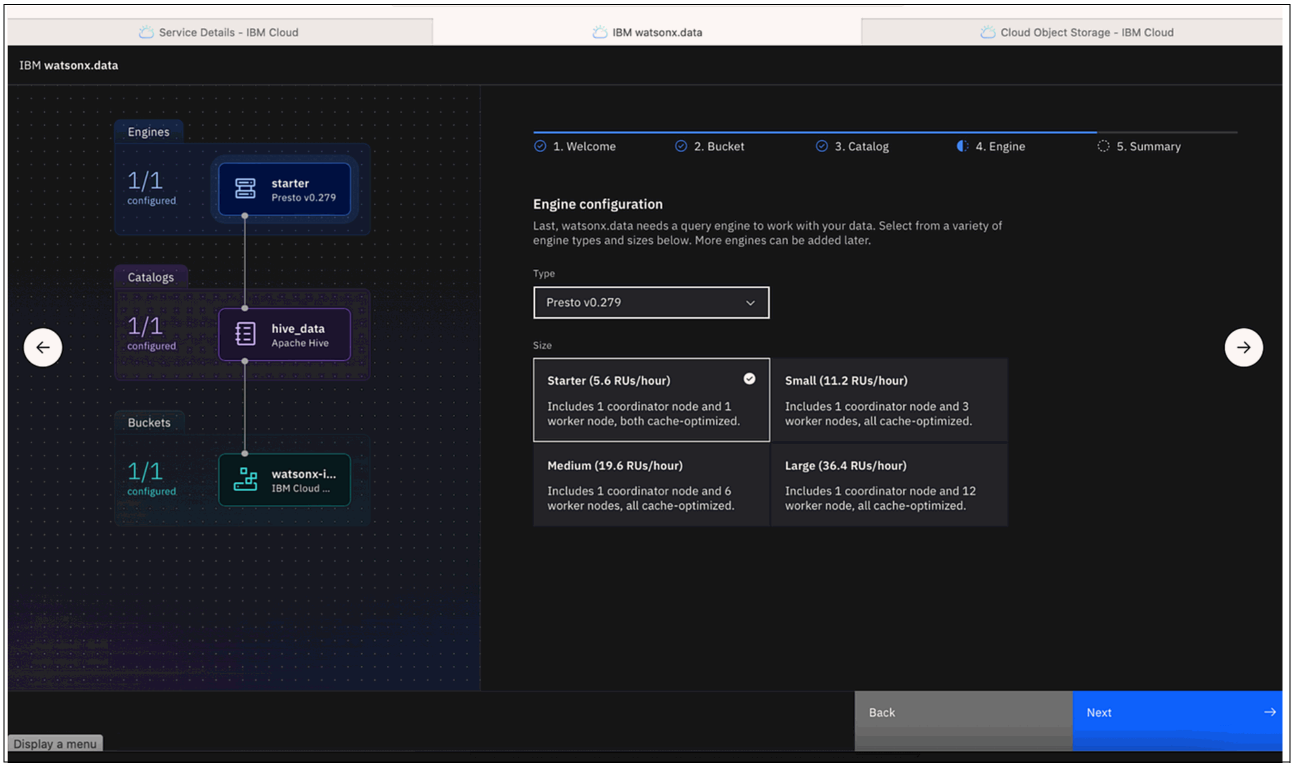


Figure 3-5 Choosing the catalog

3.4.4 Choosing the query engine

In this scenario, choose **Presto** as the query engine, and use its default options, as shown in Figure 3-6 on page 39.



Engine configuration

Last, watsonx.data needs a query engine to work with your data. Select from a variety of engine types and sizes below. More engines can be added later.

Type

Presto v0.279

Size

Starter (5.6 RUs/hour) ✓

Includes 1 coordinator node and 1 worker node, both cache-optimized.

Small (11.2 RUs/hour)

Includes 1 coordinator node and 3 worker nodes, all cache-optimized.

Medium (19.6 RUs/hour)

Includes 1 coordinator node and 6 worker nodes, all cache-optimized.

Large (36.4 RUs/hour)

Includes 1 coordinator node and 12 worker node, all cache-optimized.

Figure 3-6 Choosing the query engine

3.4.5 Data ingestion through Spark and Query by using Presto

For simplicity, assume that both buckets are within the same IBM Cloud Object Storage instance. However, these buckets can be in different IBM Cloud Object Storage instances or even on Amazon EMR S3 storage.

- `lh-xxx`: The bucket that you mapped while setting up the catalog-bucket in 3.4.2, “Configuring an S3 IBM Cloud Object Storage bucket” on page 37.
- `lh-xxx-data`: The bucket that contains the data sets to read from and create tables into the lakehouse.

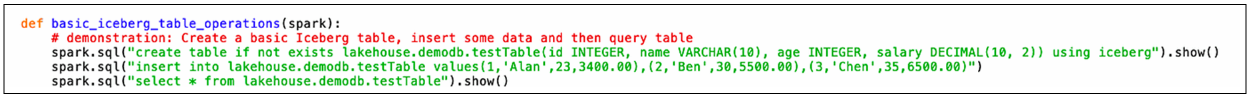
To upload the test data to the designated bucket, see Getting started with Spark use case.

Creating a database or schema in IBM watsonx.data

By using Spark, establish a new schema within the previously configured catalog. Then, create multiple tables within this schema and load them with the necessary data.

Data ingestion by using INSERT: Creating a simple table and inserting data

Note the clause that uses `iceberg`. Iceberg is the table format that is specified, as shown in Figure 3-7.



```
def basic_iceberg_table_operations(spark):
    # demonstration: Create a basic Iceberg table, insert some data and then query table
    spark.sql("create table if not exists lakehouse.demodb.testTable(id INTEGER, name VARCHAR(10), age INTEGER, salary DECIMAL(10, 2)) using iceberg").show()
    spark.sql("insert into lakehouse.demodb.testTable values(1,'Alan',23,3400.00),(2,'Ben',30,5500.00),(3,'Chen',35,6500.00)")
    spark.sql("select * from lakehouse.demodb.testTable").show()
```

Figure 3-7 Iceberg is the table format that is specified

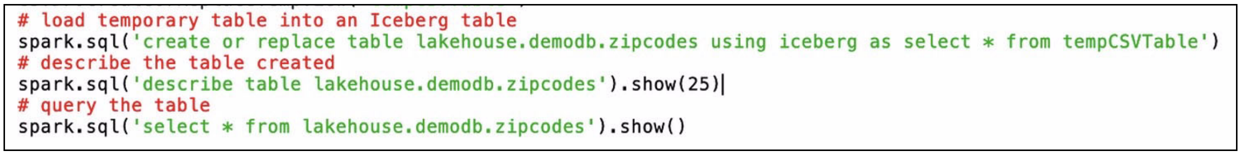
Data ingestion of the Parquet data: Creating a Parquet format table that is managed by the Iceberg table format

In this example, you use the Iceberg table format and Parquet as the data format.

Read data from the designated test bucket and write it back to the lakehouse bucket (`lh-xxx-data` in this example).

Data ingestion of the CSV data: Creating a CSV format table that is managed by the Iceberg table format

Read a CSV file from the test bucket and create an Iceberg table by using the Create Table As (CTAS) statement, as shown in Figure 3-8.

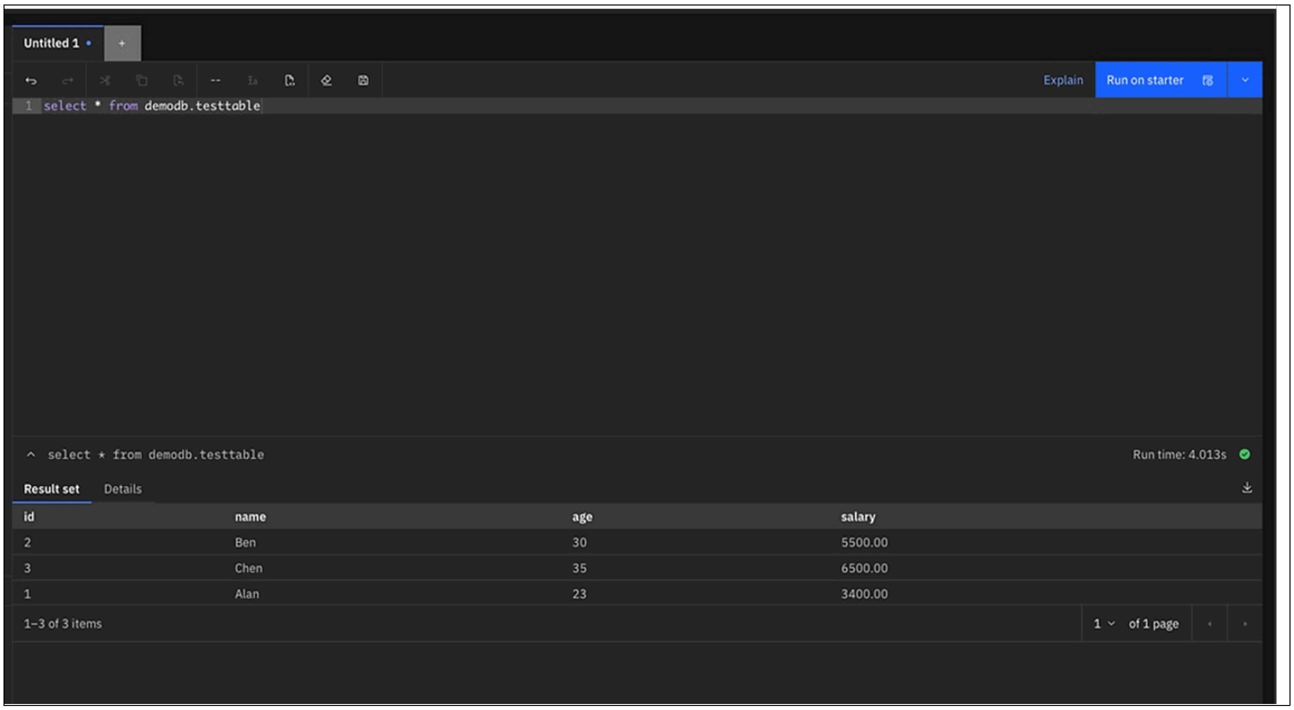


```
# load temporary table into an Iceberg table
spark.sql('create or replace table lakehouse.demodb.zipcodes using iceberg as select * from tempCSVTable')
# describe the table created
spark.sql('describe table lakehouse.demodb.zipcodes').show(25)
# query the table
spark.sql('select * from lakehouse.demodb.zipcodes').show()
```

Figure 3-8 Creating a Parquet format table that is managed by the Iceberg table format

Querying the data from Presto

This view that is shown in Figure 3-9 presents all the tables that you created by using the Spark application within the associated catalog. These tables can now be queried through the IBM watsonx.data Query interface.



```
select * from demodb.testtable
```

id	name	age	salary
2	Ben	30	5500.00
3	Chen	35	6500.00
1	Alan	23	3400.00

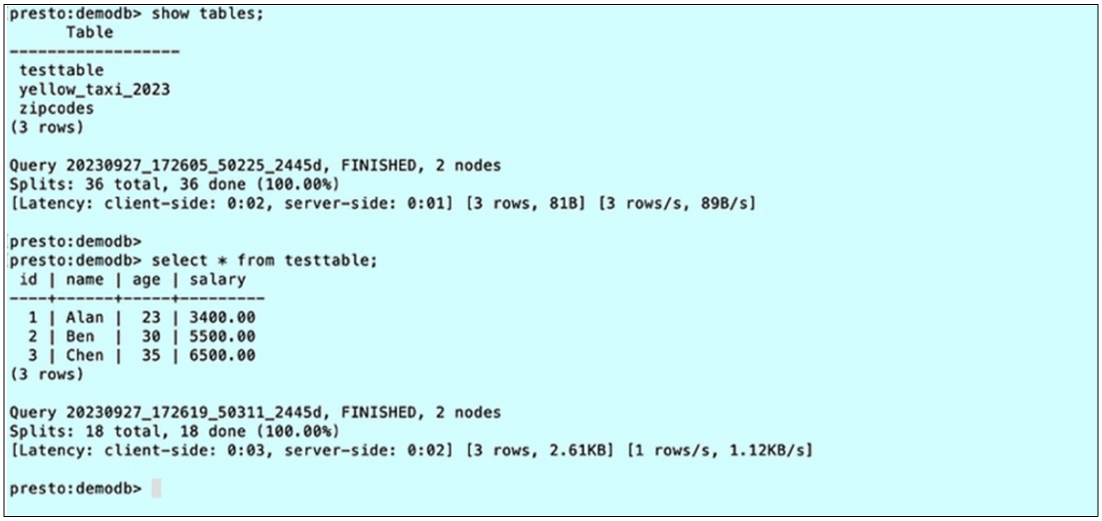
Figure 3-9 Querying the data from Presto

3.4.6 Querying from the IBM watsonx.data Query Workspace

You can query tables from a stand-alone Presto CLI by running the following command:

```
./presto\
 --server <https://<watsonx.data_MetastoreHost:port> \
 --catalog iceberg_data \
 --schema default \
 --user ibmlhapikey_<your-username> \
 --password
```

Figure 3-10 shows the output of this query.



```
presto:demodb> show tables;
      Table
------------------
 testtable
 yellow_taxi_2023
 zipcodes
(3 rows)

Query 20230927_172605_50225_2445d, FINISHED, 2 nodes
Splits: 36 total, 36 done (100.00%)
[Latency: client-side: 0:02, server-side: 0:01] [3 rows, 81B] [3 rows/s, 89B/s]

presto:demodb>
presto:demodb> select * from testtable;
 id | name | age | salary
----+------+-----+---------
  1 | Alan |  23 | 3400.00
  2 | Ben  |  30 | 5500.00
  3 | Chen |  35 | 6500.00
(3 rows)

Query 20230927_172619_50311_2445d, FINISHED, 2 nodes
Splits: 18 total, 18 done (100.00%)
[Latency: client-side: 0:03, server-side: 0:02] [3 rows, 2.61KB] [1 rows/s, 1.12KB/s]

presto:demodb>
```

Figure 3-10   Output of this query

3.4.7 Querying from the Presto CLI

Once the data ingestion process is complete, you can confirm success by verifying the presence of both data files and their corresponding metadata files within the IBM watsonx.data catalog's associated bucket.

3.5 Data pipeline considerations for open lakehouse

IBM StreamSets infuses intelligent apps with the power of streaming data, and data that is streamed intelligently.

StreamSets provides real-time data ingestion at scale so that you can deploy reliable, smart streaming data pipelines across hybrid cloud environments at scale. StreamSets pipelines stream structured, semi-structured, and unstructured data from any source. StreamSets also automatically detects changes in data structures and schemas and sends alerts about them, so you can seamlessly adapt to changing business requirements with zero downtime.

These intelligent data pipelines also adapt to unexpected data structural shifts with drag-and-drop, pre-built processors to automatically identify and adapt to data drift. The net effect is to substantially enhance your real-time decision-making and reduce the risks that are associated with data flow across your organization.

SingleStore with IBM® is a high-performance database that is designed to deliver millisecond-level insights on massive data sets. Its unique architecture and features make it an ideal foundation for intelligent applications:

- Unified data platform: The SingleStore Universal Storage engine seamlessly combines row-store and column-store technologies to enable efficient execution of both online transactional processing (OLTP) and online analytical processing (OLAP) workloads on a single platform. This approach eliminates the need for separate databases and complex ETL processes, which streamlines operations and reduces costs.
- Real-time data ingestion: SingleStore Pipelines offer high-speed data ingestion from various sources, including Kafka, S3, and HDFS. Combined with the core database engine, this approach helps ensure rapid query response times, even for complex queries against large data sets.
- Scalability and flexibility: SingleStore horizontal scalability and separation of storage and compute enable cost-effective performance optimization. Workspaces provide isolated compute environments for different workloads, helping to ensure optimal resource allocation and low-latency access to shared data. Also, SingleStore supports a wide range of data models, including relational, vector search, full-text search, time-series, geospatial, JSON, and BSON.

By using SingleStore powerful capabilities, organizations can unlock the full potential of their data, drive innovation, and gain a competitive edge.

3.5.1 IBM watsonx.data: Simplifying data for AI

IBM watsonx.data streamlines data management for AI applications in the following ways:

- Unifying data: Consolidating data from various sources into a single, unified view
- Optimizing performance: Enhancing data performance and reducing costs
- Enabling real-time insights: Integrating with tools like StreamSets and SingleStore for real-time data processing and analysis
- Powering AI: Preparing data for AI models and supporting deep learning

By simplifying data management and enabling real-time insights, IBM watsonx.data empowers organizations to harness the full potential of AI, as shown in Figure 3-11.

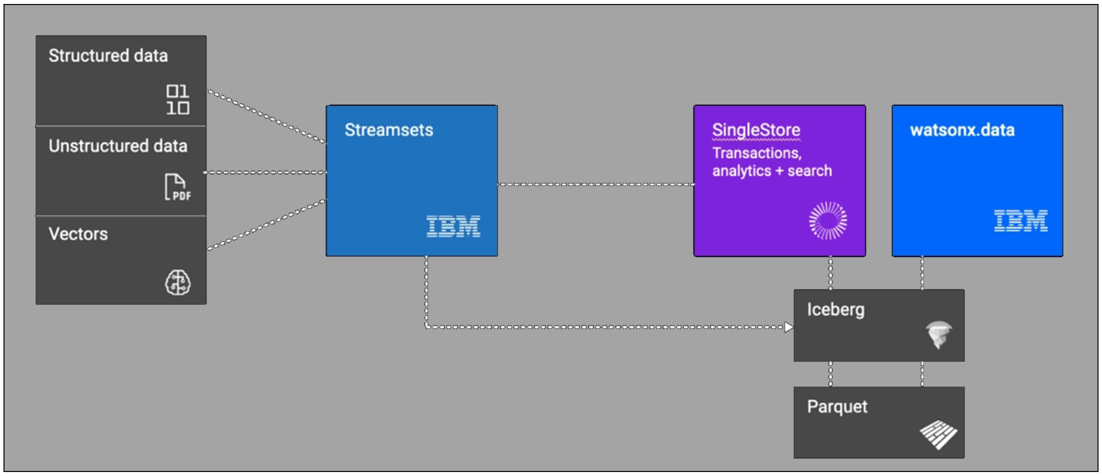


Figure 3-11 A real-time intelligent platform

3.5.2 DataStage ingestion of data into IBM watsonx.data

This section explores how to use IBM DataStage to seamlessly ingest data into IBM watsonx.data by highlighting the benefits of using the DataStage powerful capabilities for large-scale data transfers and streamlined access within your IBM watsonx.data environment.

Wide connectivity support

DataStage offers seamless data access from various sources, which include on-premises databases, cloud storage, SaaS applications, and more. Users can utilize over 60 native connectors or establish custom connections for enhanced flexibility.

Data can be loaded into a cloud storage bucket for initial staging. At the time of writing, data transfer to IBM watsonx.data requires a notebook script. However, a future-optimized connector will enable direct, native integration for efficient data loading.

Superior performance with parallel processing

DataStage uses parallel processing to ingest data faster. It can automatically or manually partition and repartition data flows for optimal performance. This approach efficiently distributes workload across resources by scaling to handle varying data volumes. By speeding up ingestion, DataStage reduces time to insight for faster analytics and decision-making.

Reducing costs while maximizing accessibility and throughput

Combining DataStage with IBM watsonx.data helps users cut data warehouse expenses. Cloud data warehouses can be expensive for storage and compute, especially with growing data and complex workloads. DataStage ingests data efficiently into IBM watsonx.data, and then uses the appropriate query engines as based on user needs. This approach centralizes data and optimizes workloads across engines and storage tiers, which minimizes data warehouse costs without sacrificing performance

Automatic detection of new files

DataStage automatically detects new data in your sources and seamlessly loads it into IBM watsonx.data. This approach eliminates manual effort and streamlines data processing, which enables faster access to insights.

3.5.3 DataStage and management of data within IBM watsonx.data

Once data is in IBM watsonx.data, DataStage can manage operational workloads or create new data pipelines.

Graphically design data pipelines

With DataStage, you can build ETL/ELT pipelines directly within IBM watsonx.data, which saves on egress costs. You can use more than 60 connectors and pre-built transformations in a drag-and-drop interface for 9x faster development compared to coding. You can switch between ETL and ELT modes seamlessly without rebuilding pipelines. You can use Apache Spark or Presto within IBM watsonx.data for diverse tasks like BI, reporting, or data science.

By combining DataStage and IBM watsonx.data, you gain the following benefits:

- Faster pipeline development: Build pipelines within IBM watsonx.data with minimal egress costs.
- Simplified workflow: Use a drag-and-drop interface for efficient pipeline creation.
- Flexibility: Switch between ETL and ELT modes.
- Powerful query engines: Use Spark or Presto for diverse data analysis needs.

Auto-responding to source schema changes

DataStage simplifies data pipeline management by automatically detecting and adapting to changes in source data schemas. This approach eliminates the need for manual adjustments, which saves time and effort. With automated schema evolution, users can deploy pipelines faster and focus on higher-value tasks.

Autoscaling to burst compute

DataStage is built to handle varying data volumes, making it ideal for businesses with seasonal or unpredictable data loads. By dynamically scaling resources, DataStage helps ensure smooth operations during peak periods, such as holiday seasons for retail companies. This proactive approach enables organizations to efficiently manage data and extract insights, regardless of the data volume.

4

Protecting data

This chapter delves into the core aspects of protecting data within IBM watsonx.data. It begins with an overview of users and groups as they relate to the lakehouse architecture, and examining how these classifications create a foundation for managing access. Next, this chapter explores defining roles and responsibilities, where you assign the appropriate permissions to various job functions, such as data scientists, analysts, and this chapter describes establishing access control lists (ACLs), which are a critical mechanism for enforcing security policies at a granular level. ACLs specify which users and groups have the right to access particular resources, which protect data assets from unauthorized access and modifications.

This chapter provides a comprehensive guide to securing data in an open lakehouse environment, which enables organizations to harness their data's potential with confidence while adhering to the highest standards of data protection and compliance.

This chapter has the following sections:

4.1 Users and groups in an open lakehouse

In IBM watsonx.data, managing users and groups effectively is fundamental to establishing a secure and collaborative open lakehouse environment. As data becomes increasingly accessible to diverse teams within organizations, defining user roles and organizing them into groups becomes essential for efficient access control and compliance. In IBM watsonx.data, users represent individual team members, such as data scientists, engineers, and analysts, each with specific roles and permissions that are tailored to their responsibilities. Grouping these users based on their functional roles enables administrators to streamline permissions management, which helps ensure that data is accessible only to those users who require it while maintaining security standards.

IBM watsonx.data facilitates this organization by enabling administrators to assign permissions and privileges to both users and groups. This approach simplifies the process of managing data access in large teams, and enables the enforcement of consistent security practices across the organization. By using user and group structures, IBM watsonx.data empowers organizations to protect their data assets, foster collaboration, and ensure compliance within the open lakehouse framework.

4.1.1 Overview of user and group management in open lakehouses

User and group management in a lakehouse environment involves defining roles, policies, and permissions that dictate who can access data and under what conditions. Effective user and group management allows organizations to accomplish the following goals:

- Control data access: Ensure that users access only the data that they are permitted to see.
- Simplify permission assignments: Use groups and roles to efficiently assign permissions to multiple users.
- Maintain auditability: Track and monitor data access for regulatory and compliance requirements.
- Enable scalability: Implement access controls that support large and diverse teams without manual overhead.

With lakehouses, where data might be semi-structured or unstructured and stored across distributed systems, maintaining this level of control and visibility requires advanced strategies, which include the use of attribute-based access control (ABAC), dynamic groups, and multi-layered access controls.

4.1.2 Business use cases for user and group management

This section delves into compelling business use cases for user and group management in IBM watsonx.data, and showcases how it can empower organizations to achieve data security, streamline access control, and foster efficient collaboration within their data analysis and AI workflows.

Attribute-based access control

ABAC is an advanced access control model that considers attributes that are associated with users, groups, and resources. Rather than assigning static permissions, ABAC enables dynamic, context-driven access based on attributes like department, project, geographic region, or time of day. ABAC is useful in lakehouse environments, where access requirements are complex and might vary significantly across the organization.

Business use case: Regulated financial services

A financial services company managing a large lakehouse for customer and transaction data must comply with regulatory requirements. Certain data sets are sensitive and need restricted access. For example, customer financial records are accessible only to users who work in specific regions and departments (for example, North American finance employees) and who have completed the required training.

Using ABAC, the company defines attributes such as Department, Region, and Training Status and creates policies that enforce these attributes. Access to sensitive data is granted only if all conditions are met, which provides flexibility in assigning access without creating an excessive number of roles.

Dynamic group membership

Dynamic group membership enables the automatic assignment of users to groups based on specific attributes or conditions. Unlike static groups, where users are manually added or removed, dynamic groups are updated in real time, which helps ensure that access rights reflect the status of each user.

Business use case: Dynamic grouping for departmental access

A large retail company that uses a lakehouse for operational analytics has different departments that require varied access to the data. For example, marketing needs access to customer analytics, and logistics requires supply chain data. Because employees move between departments or take on new roles, their access rights must change dynamically.

By implementing dynamic groups based on attributes such as Department, Job Title, and Location, the retail company can automatically update group memberships and permissions. When an employee changes departments, their group memberships and their access to data are updated without manual intervention.

Hierarchical access models and group inheritance

Hierarchical access models use a parent-child structure to organize users and groups. This approach is useful in lakehouses where there are nested levels of data sensitivity or user roles, and it enables administrators to apply access controls to a top-level group that cascade to subgroups.

Business use case: Multinational corporation with regional data access

A multinational corporation managing customer data in a lakehouse requires regional access control for different legal jurisdictions. Although regional teams need access to data relevant to their area, global executives require access across all regions.

By using a hierarchical model, the corporation sets up a parent group for global data access and creates regional child groups under it. Permissions that are assigned to the global group are inherited by regional teams, but regional policies restrict their access to specific regional data. This hierarchical structure helps ensure consistent access while supporting compliance with local regulations.

4.1.3 Implementing user and group management in open lakehouses

This section describes strategies for implementing effective access controls in open lakehouses to help ensure data security, collaboration, and governance.

Best practices for defining user attributes

When setting up ABAC and dynamic groups in a lakehouse, carefully selected attributes are essential for reliable and scalable access control. Here are some recommended attributes:

- Role or Job Title: Specifies the user's role within the organization.
- Department or Business Unit: Controls access by organizational divisions.
- Project ID: Links users to specific project-based access requirements.
- Location: Segments access by region or jurisdiction.

Using standardized attribute names and values across systems helps ensure consistency and enables cross-functional access control.

Building policies for ABAC

For effective ABAC implementation, establish clear policies that reflect your organization's requirements. Policies should have the following characteristics:

- Atomic: Ensure that each policy addresses one specific condition (for example, Department == "Finance").
- Composable: Combine multiple policies to create complex access rules.
- Auditable: Make policies easy to review and update as requirements change.

Managing group membership and hierarchical structures

Dynamic and hierarchical group structures reduce administrative complexity and help enable the lakehouse to scale as new users join, projects evolve, or regulations change. Automated tools for creating and maintaining these groups help ensure that group memberships reflect real-time user statuses and requirements.

Challenges and considerations

While user and group management offers many benefits, it also poses challenges in complex environments. Consider the following items:

- Data sensitivity and compliance: Implement strict policies for sensitive data to comply with industry regulations (for example, GDPR and HIPAA).
- Scalability: Ensure that attribute-based policies are scalable and can handle high volumes of users and attributes.
- Policy management: Regularly audit and refine policies to maintain security as the organization evolves.

4.1.4 Overview of user and group capabilities in IBM watsonx.data

IBM watsonx.data offers extensive user and group management capabilities to enable administrators to control access based on roles, groups, and permissions that are tailored to different stages in the data pipeline. Here are some of the key features:

- Role-based access control (RBAC): Enables administrators to assign roles based on job functions to help ensure that users can access only the data and tools that they need.
- Granular permissions: Offers fine-grained control over specific actions, such as data ingestion, transformation, querying, and model deployment.
- Attribute-based access control (ABAC): Enables flexible, context-based access policies that can dynamically adjust permissions.
- Group management: Supports dynamic and hierarchical groups to simplify the management of users with similar permissions and streamline access control at scale.

IBM watsonx.data user and group management capabilities are designed to handle the diverse needs of data lakehouse environments by supporting both technical and business-focused roles.

4.1.5 Implementing group-based access in IBM watsonx.data

This section describes the benefits of managing access by user groups, which include improved data security, streamlined collaboration, and simplified administration. This approach helps ensure that your team members have the appropriate level of access to use IBM watsonx.data effectively.

Configuring roles and groups

IBM watsonx.data enables administrators to configure roles that can be assigned to specific groups. These roles encapsulate the permissions that are needed for each function, which streamlines access management across different user categories. When configuring roles, perform the following actions:

- Define role requirements: Outline the specific permissions that are needed for each group to ensure that the permissions align with user responsibilities.
- Assign users to groups: Place users in pre-configured groups that match their roles and responsibilities for quick onboarding and access updates.
- Monitor group memberships: Regularly review group memberships to ensure that users retain only the necessary permissions based on their current roles.

Integrating attribute-based access control

In environments where roles and permissions must change dynamically, IBM watsonx.data supports ABAC to accommodate context-specific access requirements. Attributes such as job title, department, or project ID can be used to accomplish the following tasks:

- Dynamically update access: Adjust permissions automatically when a user's status or role changes.
- Enhance security: Limit access to sensitive data by defining attributes that are required for specific data sets.
- Streamline administration: Reduce manual role assignments, especially for temporary or project-based access needs.

Audit and compliance

For regulatory and internal compliance, IBM watsonx.data provides audit capabilities that enable administrators to track access patterns and permissions changes. Here are some key features:

- Access logging: Log all access events for auditing purposes.
- Regular audits: Schedule audits to review and adjust group permissions as needed.
- Policy-based alerts: Set up alerts for unauthorized access attempts to help ensure that administrators are informed of potential security issues.

In Figure 4-1, you can create access groups, or give access to a trusted profile, user, or service ID access to any of the target and specific permissions as depicted.

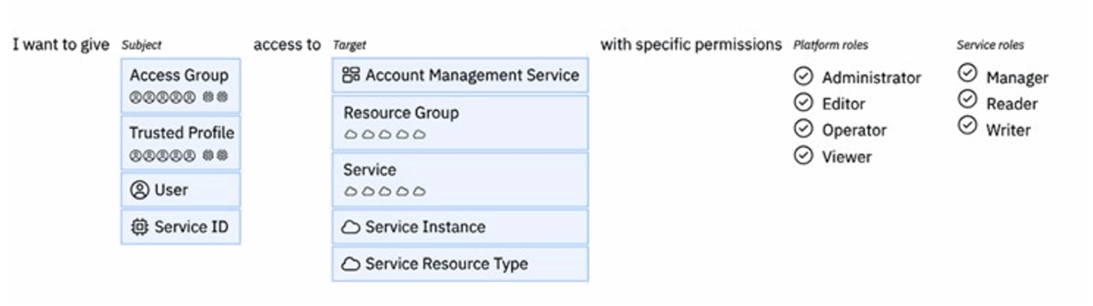


Figure 4-1 Implementing group-based policies and permissions

4.2 Defining roles and responsibilities

In a robust data lakehouse environment, defining roles and responsibilities is crucial for managing access to data and helping ensure security, compliance, and efficient workflows. This section presents the architectural design of Role-Level Access Control (RLAC) within IBM watsonx.data. It outlines the primary role types - platform roles, instance roles, and resource-level roles - and explains how they can be effectively mapped and managed. Furthermore, it discusses the integration of Single Sign-On (SSO) with these roles to facilitate seamless user access across the platform.

By establishing clear roles and responsibilities, organizations can grant users precise access to the tools and data that they need, which minimizes risk and optimizes operational efficiency.

4.2.1 The architectural design for RLAC

RLAC in IBM watsonx.data is designed to manage access at different levels within the platform so that administrators can define specific roles and assign them to users or groups. This architecture is built to handle complex access needs by combining platform roles for administrative access, instance roles for managing individual data environments, and resource-level roles for data-specific permissions.

The architectural framework of RLAC in IBM watsonx.data typically includes the following components:

- Platform layer: At the top level, platform roles are defined for administrative users who need oversight and control over the entire IBM watsonx environment.

- These roles include permissions for configuring the system, managing user accounts, and handling high-level tasks that impact the whole platform.
- Instance layer: The instance layer includes roles that are specific to individual data instances or projects within the lakehouse. Instance roles enable granular control of access to particular environments without impacting the broader platform.
- Resource layer: At the finest level, resource-level roles control access to specific data sets, tables, or resources within an instance. These roles are essential for managing access to sensitive data or defining permissions for specific data sets within the lakehouse.

By using these three layers in tandem, IBM watsonx.data enables organizations to implement a structured, scalable access control model that meets both security and operational needs. This layered approach to role management simplifies access assignments, minimizes administrative overhead, and enhances data governance.

4.2.2 Platform roles and instance roles in IBM watsonx.data

IBM watsonx.data provides two primary role types: *platform roles* and *instance roles*. They govern access at different scopes within the lakehouse environment.

Platform roles

Platform roles in IBM watsonx.data grant permissions that apply across the entire platform. These roles provide high-level administrative control over users, configurations, and settings. Platform roles are typically assigned to IT administrators or platform owners who are responsible for managing IBM watsonx.data.

Here are the platform roles:

- Administrator role: Grants comprehensive administrative access so that users can configure and manage resources across the platform. Common use cases include managing users and defining security policies.
- Editor role: Provides permissions to modify resources but not manage user roles or platform-wide configurations.
- Viewer role: Enables read-only access, which is useful for monitoring and auditing activities.
- Operator role: Focused on operational management, such as managing jobs and workflows.

These roles are typically managed through IBM Cloud Identity and Access Management (IAM) tools, which streamline role assignment and enforce compliance requirements across the platform

Instance roles

Instance roles operate at the scope of a specific IBM watsonx.data deployment or resource, which provide more granular control over data and compute resources.

Here are the instance roles:

- Metastore Admin: Full access to metadata repositories and configuration, which is critical for administrators managing catalogs in services like Db2 or Netezza.
- Metastore Viewer: Read-only access to metadata, which is ideal for users who need insights without modification capabilities.

- Data Access role: Designed primarily for service-to-service integrations, such as enabling data profiling and analytics workflows.

Instance roles help ensure that users interact only with the specific resources that are necessary for their tasks, which enhance data security and operational efficiency.

Table 4-1 shows the instance roles privileges.

Table 4-1 Instance roles privileges

Create Presto (Java)	Admin	User
Restart the internal HMS	X	
Unregister any storage	X	
Unregister any DB connection	X	
Activate cataloged buckets (restart HMS)	X	
Register and unregister own storage	X	X
Register and unregister own DB connection	X	X
Access the metastore	x	
Read access to HMS API	X	

4.2.3 Resource-level roles and permissions

Resource-level roles in IBM watsonx.data provide the most granular control over access so that administrators can define permissions on individual data sets, tables, and resources. These roles are essential for protecting sensitive data within the lakehouse and helping ensure that users access only data that is relevant to their role.

Key resource-level roles

Resource-level roles control access to individual data resources within an instance. They are crucial for enforcing data privacy and helping ensure that sensitive data is accessible only to authorized users. Here are the key resource-level roles:

- Resource Owner: Has full control over a specific resource, such as a table or data set. The resource owner can define who has access to the resource, assign permissions, and manage the lifecycle of the data.
- Resource Editor: Can modify data within the resource, which makes it suitable for users who need to update data sets, transform data, or manage content. Data engineers and data developers often take on this role for specific data sets.
- Resource Viewer: Provides read-only access to data within the resource. This role is ideal for analysts, data scientists, or business users who need access to data for reporting or analysis without modifying it.

Permissions for resource-level roles

Resource-level roles are assigned with specific permissions that govern what actions users can take on data resources. Here are the key permissions for resource-level roles:

- Read Permission: Enables users to view data within the resource.
- Write Permission: Enables users to update data, make modifications, and manage data transformations.
- Execute Permission: Grants the ability to run jobs or workflows that are associated with the resource. This permission is commonly used in ETL processes or data transformations.
- Manage Permissions: Enables users to set permissions for other users. This permission is suitable for resource owners or administrators.

By combining resource-level roles with platform and instance roles, IBM watsonx.data enables administrators to create a highly flexible access control model, which helps ensure that users have the appropriate level of access based on their responsibilities while safeguarding sensitive data from unauthorized access.

4.2.4 Best practices for resource-level role management

To maximize the effectiveness of resource-level roles, consider the following best practices:

- Adopt the Principle of Least Privilege: Grant users only the minimum level of access that they need to perform their roles. This approach minimizes the risk of unauthorized data access and helps maintain data security.
- Use Dynamic Groups for Resource Assignment: Where possible, assign resource roles that are based on dynamic groups or attributes (for example, department or role), allowing permissions to adjust automatically as users' responsibilities change.
- Regularly Audit Access Permissions: Periodically review resource-level permissions to ensure that access levels are appropriate and in compliance with security policies.
- Monitor Sensitive Data Access: For highly sensitive resources, configure alerts to monitor access patterns and identify potential security risks. This best practice is especially relevant in regulated industries where data access must be auditable.

4.3 Establishing ACLs

In IBM watsonx.data, access control lists (ACLs) are used to manage and restrict access to the data in object store and in federated databases, which helps ensure that only authorized users or services can interact with the data. ACLs are critical for maintaining security and compliance in a data environment by specifying who can read, write, or modify data, and by controlling administrative tasks.

The typical types of ACLs for data access in IBM watsonx.data include role-based ACL and policy-based ACL.

4.3.1 Role-based ACL

Roles define sets of permissions that can be assigned to users and user groups. This approach simplifies the management of permissions by grouping common access requirements under a role. Here are some examples of these roles:

- Catalog Admin: Full access to create, modify, and delete data (schemas and tables) in a particular catalog, and grant or revoke permissions for other users and groups.
- Storage Writer: Read/write access on the bucket in a particular object store.

- Milvus Viewer: Read-only access on the collections and partitions in a particular Milvus service.

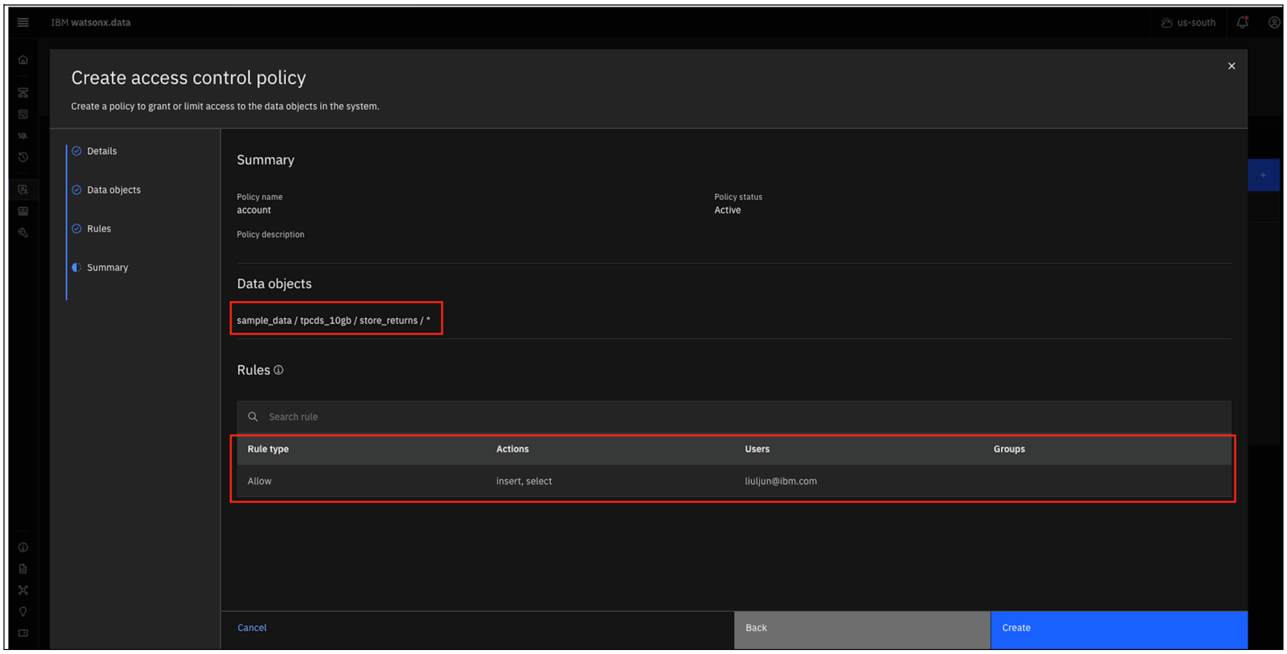
For more information about permissions on predefined roles, see the Managing roles and privileges.

4.3.2 Policy-based ACL

Policies define a set of rules that can be applied to user and group access. The rules enable more granular access control and automation of access management. Usually, a policy contains the following items:

- Subject: The data objects that the access is being requested for.
- Rules: Define the behavior of the access control. It contains the following items:
  - Type: Usually allow or deny. Some advanced types such as data masking and row-level filtering are supported by external policy engines.
  - Principal: The entity (user or group) that is requesting access.
  - Action: The operation that is requested (select, insert, delete, and others).

In Figure 4-2, the policy contains a single rule to grant “select” and “insert” permissions on the table `store_returns` under the schema `tpcds_10gb` in the catalog `sample_data` to the user liuljun@ibm.com.



Create access control policy

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

Summary

Policy name: account

Policy status: Active

Policy description

Data objects

sample_data / tpcds_10gb / store_returns / *

Rules ⓘ

Search rule

Rule type	Actions	Users	Groups
Allow	insert, select	liuljun@ibm.com	

Figure 4-2 Create an access control policy: Add a rule

IBM watsonx.data supports access control policies for several types of resources. The data objects and the corresponding rules are different. Table 4-2 describes them in detail.

Table 4-2 Access control policies

Resource	Data object	Action
Catalog	Schema/Table/Column	Select, Insert, Update, Alter, and others

Resource	Data object	Action
Storage	Folder/File or regular expression for the S3 path (`s3://<bucket-name>/<object-key>`)	Read, Write, or Delete
Milvus service	Database/Collection/Partition	ListDatabase, ListCollection, ListPartition, Search, and others

For more information about how to manage the access control policies, see Managing data policy rules.

4.3.3 Best practices to manage ACLs

To manage role-based ACLs and policy-based ACLs effectively in IBM watsonx.data and avoid overlaps, it is a best practice to establish a clear strategy that uses each method for its strengths. Here are some best practices.

Defining roles first

Roles should define broad access categories based on a user or group's function within the organization. Once these roles are established, policy-based ACLs can be used to fine-tune access for specific use cases or granular control.

Consider a typical example: Managing access to AWS S3 storage. An organization might define three user groups:

- Admins: Manage the storage lifecycle.
- Analysts: Require read-only access to files.
- Data Engineers: Need write access to upload and update files.

To implement these groups, assign roles to them:

Storage Admin: Granted to the Admin group.
Storage Reader: Granted to the Analyst group.
Storage Writer: Granted to the Data Engineer group.

If a specific analyst, such as Tom, needs temporary write access to a file, a policy-based ACL can be created to grant him that permission for a specific period. This approach allows for flexibility without compromising overall security. Importantly, Tom retains his usual read-only access to other files in the storage.

Using role hierarchies for simplified management

IBM watsonx.data uses RBAC with role hierarchies. This approach simplifies permission management by enabling higher-level roles to inherit permissions from lower-level roles, which reduce redundancy. Here is the role hierarchy:

- Storage Reader: The lowest level, which grants read-only access.
- Storage Writer: Inherits Storage Reader permissions and adds write access.
- Storage Admin: Inherits Storage Writer permissions and gains full control, which includes unregistering storage, updating properties, and managing access.

Platform-level roles also inherit from resource-level roles. For example, a platform admin inherits “view” and “remove” permissions for all resources, so they can manage the entire platform.

Avoiding redundant policies for common access

To streamline access management and reduce complexity, avoid creating multiple overlapping policies for common access needs. Redundant policies can lead to unintended consequences, such as conflicting permissions or difficulties in auditing and troubleshooting.

For example, consider a scenario where three policies are created:

- Policy A: Grants read access to the entire “example-bucket.
- Policy B: Grants read access to the “example-bucket/reports/*” folder.
- Policy C: Grants read access to the “example-bucket/finance/*” folder.

In this case, Policies B and C overlap with Policy A, which creates unnecessary complexity.

Best practice: Consolidating policies

To simplify management, consolidate overlapping policies into a single, unified policy. For example, you could create a policy, Policy D, that grants read access to both the `example-bucket/reports/` and `example-bucket/finance/` folders. This streamlined approach reduces the risk of errors, improves security, and makes it easier to manage access permissions.

4.3.4 Summary

IBM watsonx.data empowers secure and organized data access through Access Control Lists (ACLs). ACLs manage permissions for object stores and federated databases by using two primary methods:

- Role-based ACLs: Assign broad permissions through predefined roles like Catalog Admin, Storage Writer, and Milvus Viewer. This method simplifies access control.
- Policy-based ACLs: Offer granular control by defining specific rules for subjects (users or groups), actions (read, write, and so forth), and principals (who have permissions).

Best practices for managing ACLs include the following ones:

- Defining roles before applying policies.
- Using role hierarchies to streamline permission management.
- Avoiding redundant or overlapping policies to minimize complexity and conflicts.

5

Querying and manipulating data and leveraging persona-specific engines

A data lakehouse is a unified data platform that seamlessly integrates the strengths of data warehouses and data lakes. As a high-performance SQL query engine, Presto enables rapid, interactive analytics on large-scale datasets. Its versatile capabilities extend to diverse use cases, such as real-time ad-hoc queries and complex ETL processes involving terabytes of data.

This chapter has the following sections:

- “Using PrestoDB or Prestissimo engine for adhoc queries” on page 60
- “Leveraging Apache Spark engine for data engineering” on page 67
- “Execute important queries using the power of traditional RDBMS with shared open lakehouse formats” on page 81

5.1 Using PrestoDB or Prestissimo engine for adhoc queries

Presto is a distributed SQL query engine designed to efficiently process massive datasets across diverse data sources. It empowers users to perform interactive, ad-hoc analytics on data residing in various systems, including Hive, AWS S3, Hadoop, Cassandra, relational databases, NoSQL databases, and proprietary data stores. By unifying access to data from multiple sources, Presto enables organizations to conduct comprehensive analytics across their entire data landscape.

Presto's distributed SQL engine architecture leverages SQL, the industry-standard language for data manipulation. This ensures broad accessibility and compatibility with existing SQL skills and tools. By adhering to the ANSI SQL standard, Presto supports a wide range of SQL commands, including SELECT, UPDATE, DELETE, INSERT, and WHERE, enabling seamless integration with diverse data sources and BI tools.

Presto’s SQL foundation ensures broad accessibility and rapid adoption. Its SQL compatibility with other databases allows for seamless migration of existing SQL queries and BI tools, requiring minimal to no modifications.

Presto's adaptable, flexible, and extensible architecture enables seamless integration with a wide range of data sources. Its plugin mechanism allows you to connect to diverse data systems, from traditional databases to modern data lakes and warehouses. A single Presto query can effortlessly combine data from multiple sources, empowering organizations to conduct comprehensive analytics across their entire data ecosystem. The vibrant Presto community provides a rich ecosystem of connectors, further expanding the platform's capabilities.

You can see the high-level architecture diagram in Figure 5-1.

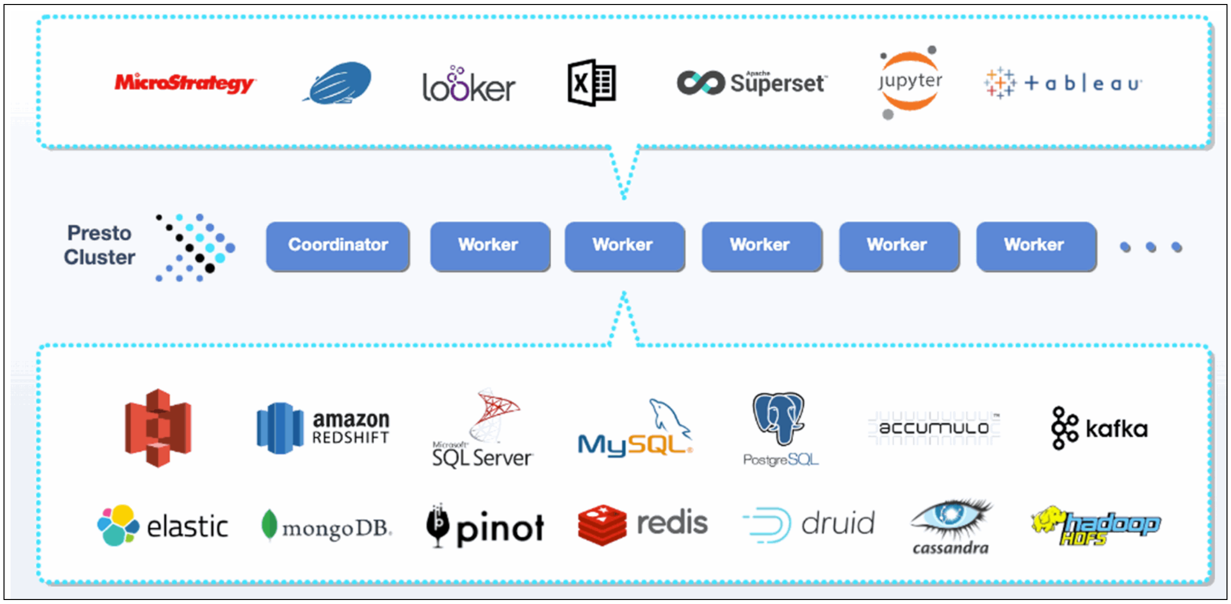


Figure 5-1 Presto high-level architecture diagram

5.1.1 Presto technical concepts

Your Presto cluster sits between your BI tools (such as Superset, Tableau and Looker) and your data sources. Presto queries across many different data sources and provides that data back to your BI tool for your organization.

A full Presto installation includes a coordinator and multiple workers. Queries are submitted from a client such as the Presto CLI to the coordinator. The coordinator parses, analyzes and plans the query execution, and then distributes the processing to the workers.

Figure 5-2 shows the Presto cluster coordinator structure.

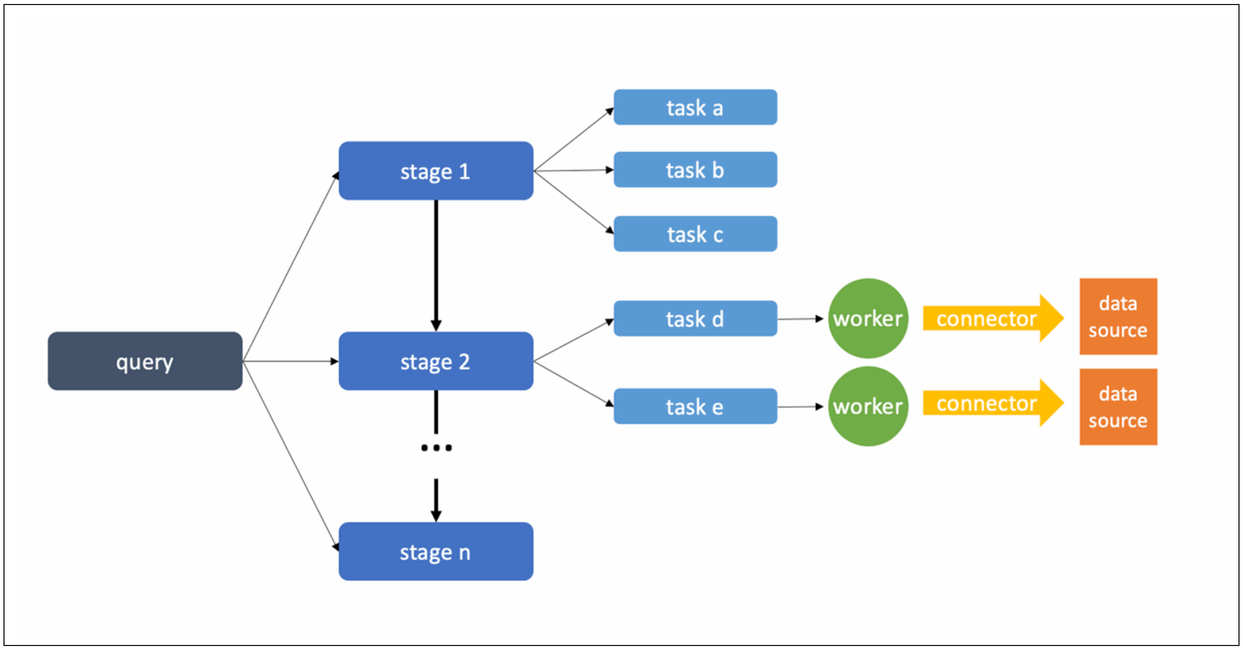


Figure 5-2 Presto cluster coordinator structure

Server types

There are three types of Presto servers: resource manager, coordinators, and workers. The following section explains the difference between them.

Resource manager

The Presto resource manager is the server that aggregates data from all coordinators and workers and constructs a global view of the cluster. A Presto installation with a disaggregated coordinator needs a resource manager. Clusters support multiple resource managers, each acting as a primary.

Coordinators and workers communicate with resource managers using a thrift API.

Coordinator

The Presto coordinator is the server that is responsible for parsing statements, planning queries, and managing Presto worker nodes. It is the "brain" of a Presto installation and is also the node to which a client connects to submit statements for execution. Every Presto installation must have a Presto coordinator alongside one or more Presto workers. For development or testing purposes, a single instance of Presto can be configured to perform both roles.

The coordinator keeps track of the activity on each worker and coordinates the execution of a query. The coordinator creates a logical model of a query involving a series of stages which is then translated into a series of connected tasks running on a cluster of Presto workers.

Coordinators communicate with workers and clients using a REST API.

Worker

After ensuring that an appropriate number of resources are available, the coordinator delegates these tasks to the worker nodes. Worker nodes process their tasks in parallel, using the relevant connector to access the underlying data source.

The connector used can vary across workers, depending on how the query was optimized and what data sources need to be accessed. As the worker nodes process their tasks, the coordinator continually monitors them using heartbeat signals. Once workers are done, the result of the tasks is sent back to the coordinator.

The coordinator can then assign workers new tasks from any remaining query stages. Once all stages are complete, the coordinator compiles the results from each stage into the final form required by the original query.

Pipelining the query stages across the network in this way ensures that any unnecessary I/O overhead is avoided. Additionally, all processing occurs in-memory, and intermediate data at the task level is stored in a buffer cache.

All of these features ensure that Presto remains extremely performant, even at petabyte sizes.

5.1.2 Data sources

There are four types of Presto data sources that you need in your deployment (Figure 5-3).

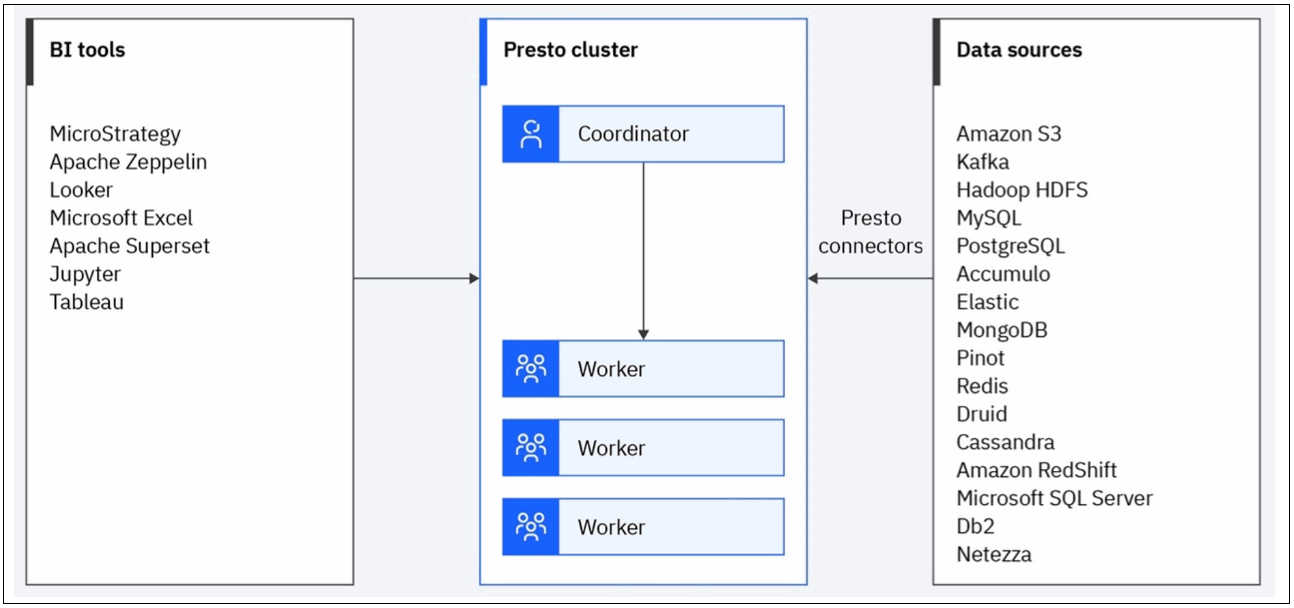


Figure 5-3 Presto cluster between BI tools and different types of data sources

Connector

A connector adapts Presto to a data source such as Hive or a relational database. You can think of a connector the same way you think of a driver for a database. It is an implementation of Presto's SPI which allows Presto to interact with a resource using a standard API.

Presto contains several built-in connectors: a connector for JMX, a System connector which provides access to built-in system tables, a Hive connector, and a TPCH connector designed to serve TPC-H benchmark data. Many third-party developers have contributed connectors so that Presto can access data in a variety of data sources.

Every catalog is associated with a specific connector. If you examine a catalog configuration file, you will see that each contains a mandatory property connector.name which is used by the catalog manager to create a connector for a given catalog. It is possible to have more than one catalog use the same connector to access two different instances of a similar database. For example, if you have two Hive clusters, you can configure two catalogs in a single Presto cluster that both use the Hive connector, allowing you to query data from both Hive clusters (even within the same SQL query).

Catalog

A Presto catalog includes schemas and refers to a data source using a connector. For example, you can configure a JMX catalog to provide access to JMX information via the JMX connector. When you run a SQL statement in Presto, you are running it against one or more catalogs. Other examples of catalogs include the Hive catalog to connect to a Hive data source.

When addressing a table in Presto, the fully-qualified table name is always rooted in a catalog. For example, a fully-qualified table name of hive.test_data.test would refer to the test table in the test_data schema in the hive catalog.

Catalogs are defined in properties files stored in the Presto configuration directory.

Schema

Schemas are a way to organize tables. Together, a catalog and schema define a set of tables that can be queried. When accessing Hive or a relational database such as MySQL with Presto, a schema translates to the same concept in the target database. Other types of connectors may choose to organize tables into schemas in a way that makes sense for the underlying data source.

Table

A table is a set of unordered rows which are organized into named columns with types. This is the same as in any relational database. The mapping from source data to tables is defined by the connector.

Figure 5-4 on page 64 shows a Presto cluster between coordinator and workers data structure.

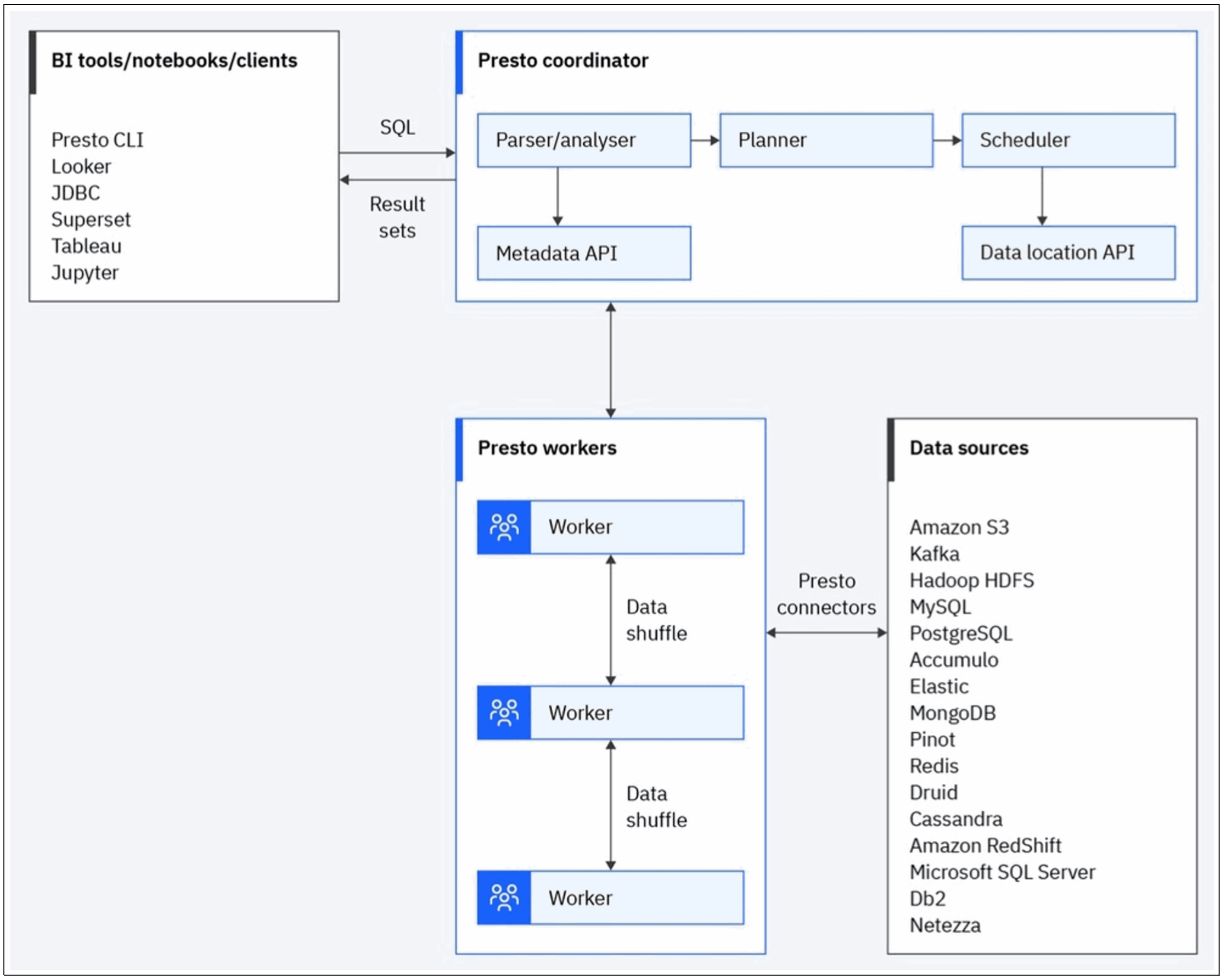


Figure 5-4 Presto cluster between coordinator and workers data structure

5.1.3 Executing a query

If your query joins together many large tables, it may need multiple stages to execute, aggregating tables together. After each execution stage, there may be intermediate data sets. Unlike distributed query engines such as Hive that were designed to persist intermediate results to disk, Presto saves time by executing queries in the memory of the worker machines. It performs operations on intermediate datasets there, instead of persisting them to disk.

With Presto, data can reside in many different places and Presto performs the executions in memory across your workers, moving data between workers as needed. This process avoids the need to write and read from disk between stages; the result: faster query execution time.

The key pieces of the query execution model are as follows:

Statement

Presto executes ANSI-compatible SQL statements. When the Presto documentation refers to a statement, it is referring to statements as defined in the ANSI SQL standard which consist of clauses, expressions, and predicates.

Some readers might be curious why this section lists separate concepts for statements and queries. This is necessary because, in Presto, statements simply refer to the textual representation of a SQL statement. When a statement is executed, Presto creates a query along with a query plan that is then distributed across a series of Presto workers.

Query

When Presto parses a statement, it converts it into a query and creates a distributed query plan which is then realized as a series of interconnected stages running on Presto workers. When you retrieve information about a query in Presto, you receive a snapshot of every component that is involved in producing a result set in response to a statement.

The difference between a statement and a query is simple. A statement can be thought of as the SQL text that is passed to Presto, while a query refers to the configuration and components instantiated to execute that statement. A query encompasses stages, tasks, splits, connectors, and other components and data sources working in concert to produce a result.

Stage

When Presto executes a query, it does so by breaking up the execution into a hierarchy of stages. For example, if Presto needs to aggregate data from one billion rows stored in Hive, it does so by creating a root stage to aggregate the output of several other stages, all of which are designed to implement different sections of a distributed query plan.

The hierarchy of stages that comprises a query resembles a tree. Every query has a root stage which is responsible for aggregating the output from other stages. Stages are what the coordinator uses to model a distributed query plan, but stages themselves do not run on Presto workers.

Task

As mentioned in “Stage” on page 65, stages model a particular section of a distributed query plan, but stages themselves do not execute on Presto workers. To understand how a stage is executed, you'll need to understand that a stage is implemented as a series of tasks distributed over a network of Presto workers.

Tasks are the *work horse* in the Presto architecture. A distributed query plan is deconstructed into a series of stages which are then translated to tasks which then act upon or process splits. A Presto task has inputs and outputs, and just as a stage can be executed in parallel by a series of tasks, a task is executing in parallel with a series of drivers.

Split

Tasks operate on splits which are sections of a larger data set. Stages at the lowest level of a distributed query plan retrieve data via splits from connectors, and intermediate stages at a higher level of a distributed query plan retrieve data from other stages.

When Presto is scheduling a query, the coordinator will query a connector for a list of all splits that are available for a table. The coordinator keeps track of which machines are running which tasks and what splits are being processed by which tasks.

Driver

Tasks contain one or more parallel drivers. Drivers act upon data and combine operators to produce output that is then aggregated by a task and delivered to another task in another stage. A driver is a sequence of operator instances. You can think of a driver as a physical set of operators in memory. It is the lowest level of parallelism in the Presto architecture. A driver has one input and one output.

Operator

An operator consumes, transforms, and produces data. For example, a table scan fetches data from a connector and produces data that can be consumed by other operators, and a filter operator consumes data and produces a subset by applying a predicate over the input data.

Exchange

Exchanges transfer data between Presto nodes for different stages of a query. Tasks write data into an output buffer and consume data from other tasks using an exchange client.

5.1.4 Prestissimo (C++ version of Presto)

Presto C++, sometimes referred to by the development name Prestissimo, is a drop-in replacement for Presto workers written in C++ and based on the Velox library. It implements the same RESTful endpoints as Java workers using the Proxygen C++ HTTP framework. Because communication with the Java coordinator and across workers is only done using the REST endpoints, Presto C++ does not use JNI and does not require a JVM on worker nodes.

Presto aims to be the top performing system for data lakes. To achieve this goal, the Presto community is moving the Presto evaluation engine from the native Java-based implementation to a new implementation written in C++ using Velox.

By moving the evaluation engine to a library, the intent is to enable the Presto community to focus on more features and better integration with table formats and other data warehousing systems.

Supported use cases

The following are supported use cases:

- Only specific connectors are supported in the Presto C++ evaluation engine.
- Hive connector for reads and writes, including CTAS, are supported.
- Iceberg tables are supported only for reads.
- Iceberg connector supports both V1 and V2 tables, including tables with deleted files.
- TPCH connector, with tpch.naming=standard catalog property.

Prestissimo general limitations

The C++ evaluation engine has a number of limitations:

- Not all built-in functions are implemented in C++. Attempting to use unimplemented functions results in a query failure.
- Not all built-in types are implemented in C++. Attempting to use unimplemented types will result in a query failure.
- All basic and structured types in Data Types are supported, except for CHAR, TIME, and TIME WITH TIMEZONE. These are subsumed by VARCHAR, TIMESTAMP and TIMESTAMPWITH TIMEZONE.
- Presto C++ only supports unlimited length VARCHAR, and does not honor the length n in varchar[n].
- The following types are not supported: IPADDRESS, IPPREFIX, UUID, KHYPERLOGLOG, P4HYPERLOGLOG, QDIGEST, TDIGEST, GEOMETRY, BINGTILE.

- Certain parts of the plugin SPI are not used by the C++ evaluation engine. In particular, C++ workers will not load any plugin in the plugins directory, and certain plugin types are either partially or completely unsupported.
- PageSourceProvider, RecordSetProvider, and PageSinkProvider do not work in the C++ evaluation engine.
- User-supplied functions, types, parametric types and block encodings are not supported.
- The event listener plugin does not work at the split level.
- User-defined functions do not work in the same way.
- Memory management works differently in the C++ evaluation engine. In particular:
- The OOM killer is not supported.
- The reserved pool is not supported.
- In general, queries may use more memory than they are allowed to through memory arbitration.
- In C++ based Presto, reduce_agg is not permitted to return null in either the inputFunctionor the combineFunction. In Presto (Java), this is permitted but undefined behavior.

Note: For practical guidance on how to use Presto refer to 3.2, “Integrating external data sources: Federation in PrestoDB” on page 30 and 8.3.1, “Gathering the required information in IBM watsonx.data” on page 119.

5.2 Leveraging Apache Spark engine for data engineering

You can use watsonx.data to seamlessly integrate with Spark engine to achieve the following use cases:

- Ingesting large volumes of data into watsonx.data tables.
- Table maintenance operations to enhance performance.
- Complex analytics workloads that are difficult to represent as queries.

5.2.1 Creating and customizing internal Spark engine inside watsonx.data

Now let us delve deeper into Native Spark. We will explore the following steps:

1. You first create a native spark engine inside your watsonx.data instance as shown in Figure 5-5 on page 68. Click **Add Engine**.

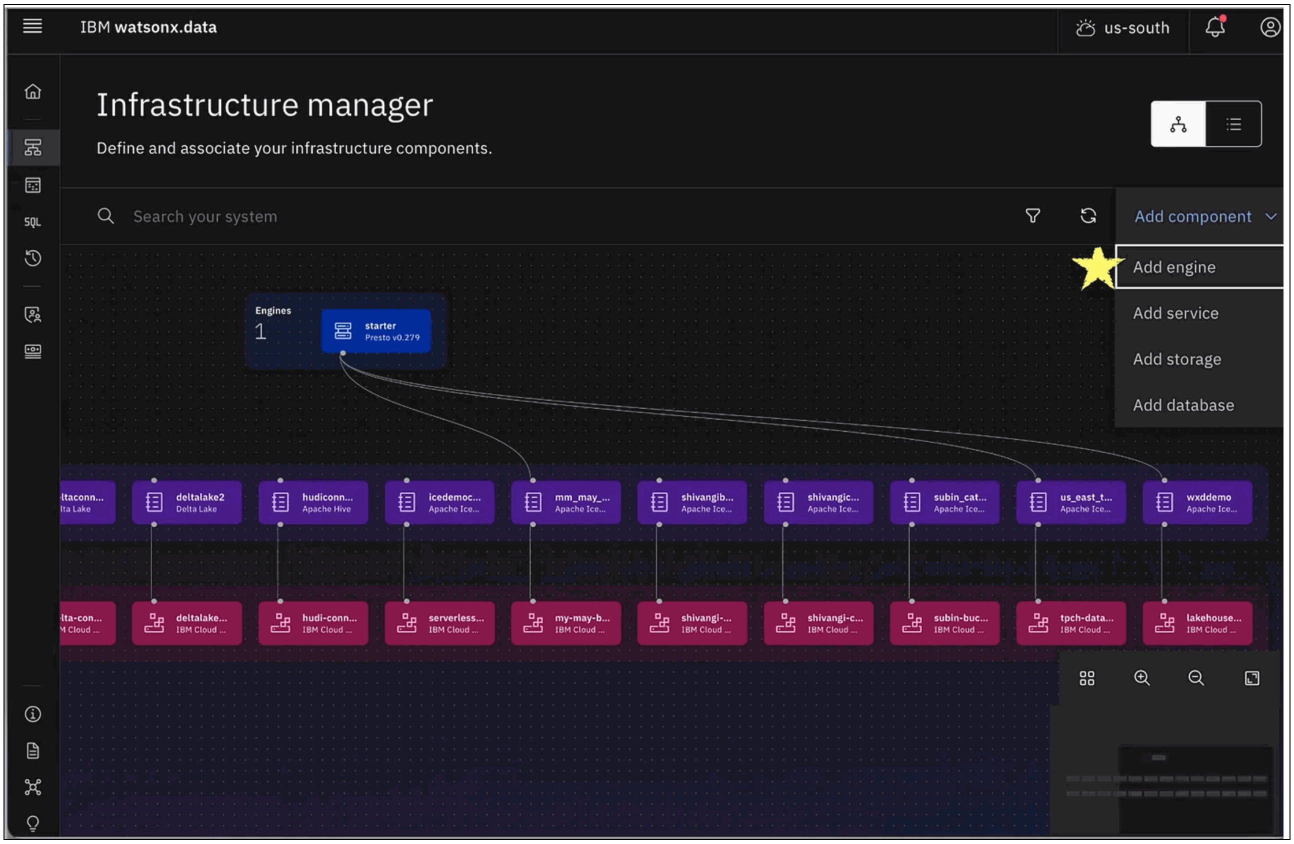
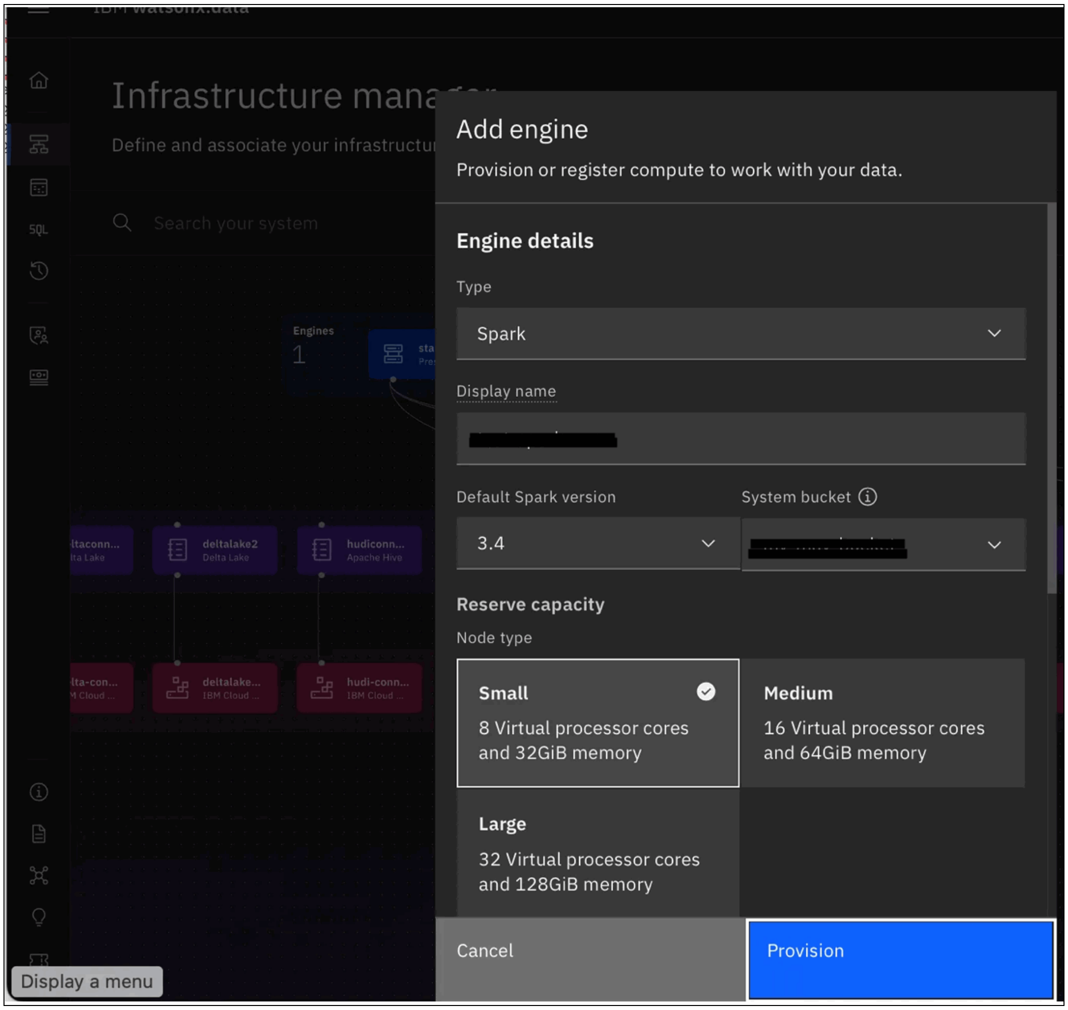


Figure 5-5 Add new native Spark engine to watsonx.data

2. Select the Type as **Spark**. Give it a name.
3. Select the default Spark runtime version. Currently watsonx.data supports two versions - Spark 3.3 and Spark 3.4.
4. Associate a **System bucket**. This bucket is the "instance home" bucket, where application logs, spark-events and other associated information are stored.
5. Choose a **Node type**. You can choose a node type (Small, Medium or Large). See Figure 5-6 on page 69.



Add engine

Provision or register compute to work with your data.

Engine details

Type: Spark

Display name: [redacted]

Default Spark version: 3.4

System bucket: [redacted]

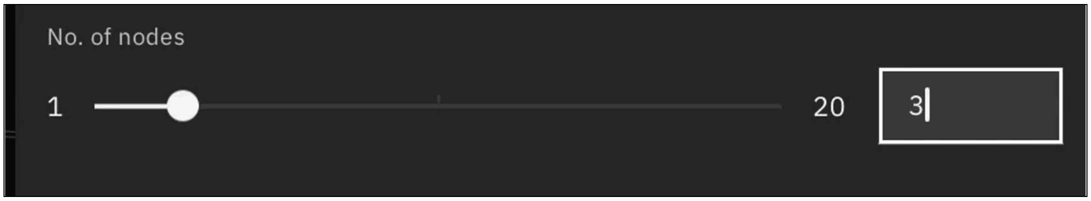
Reserve capacity

Node type

- ☑ **Small** 8 Virtual processor cores and 32GiB memory
- **Medium** 16 Virtual processor cores and 64GiB memory
- **Large** 32 Virtual processor cores and 128GiB memory

Figure 5-6 Internal Spark engine details

6. Choose the number of nodes. This is the number of nodes that you want reserved (up to 20 nodes) for your Spark engine. See Figure 5-7.



No. of nodes: 3

Figure 5-7 Number of node configurations

7. The billing for watsonx.data is based on fixed base charge plus the number of RUs (Resource Units) consumed. To understand how the billing is done for the Spark engine, refer to the **About** tab at `https://cloud.ibm.com/watsonxdata`.

8. Next, you can then choose to associate one or more existing catalogs with the engine. In Figure 5-8 on page 70 we paired two catalogs with a single Spark engine. This demonstrates that a Spark engine, like a Presto engine or Milvus service, can be paused when inactive. Resume can take up to 10 minutes.

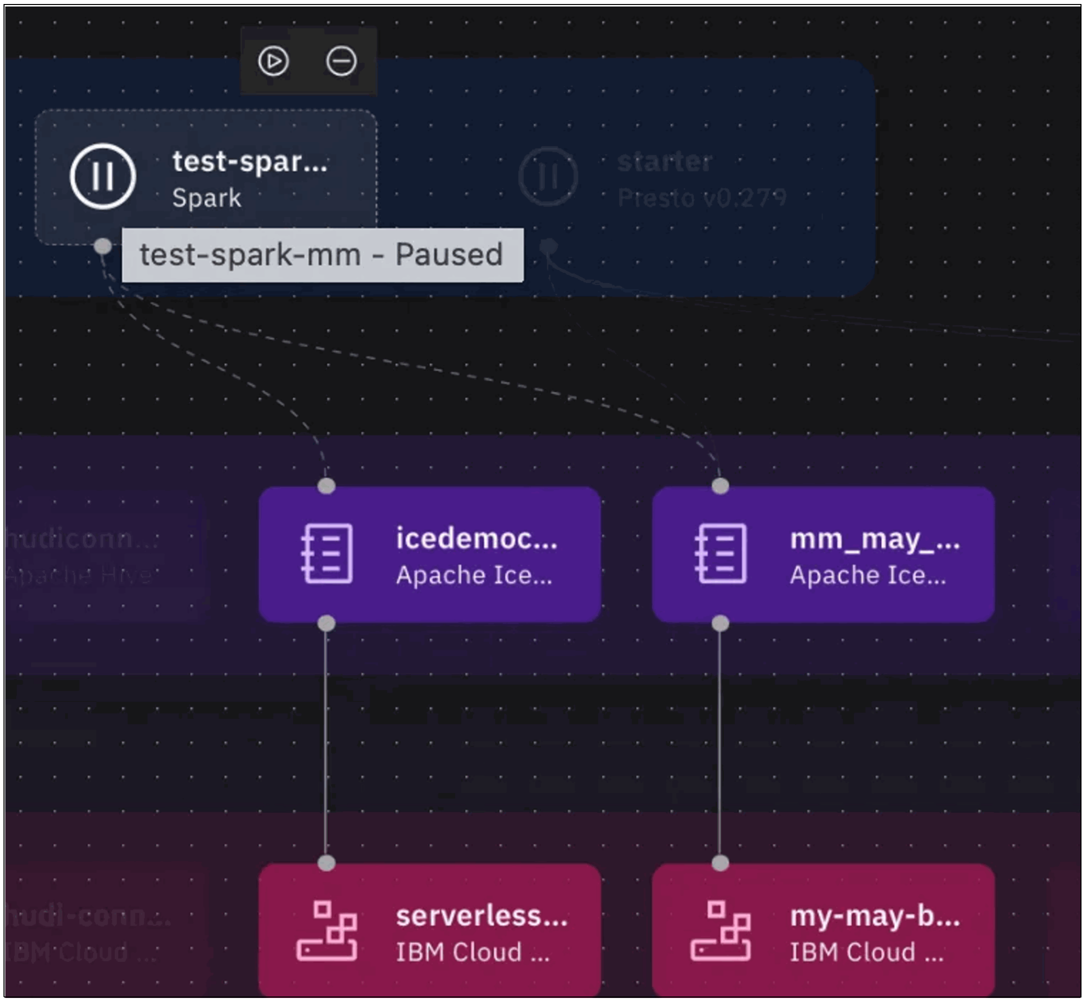
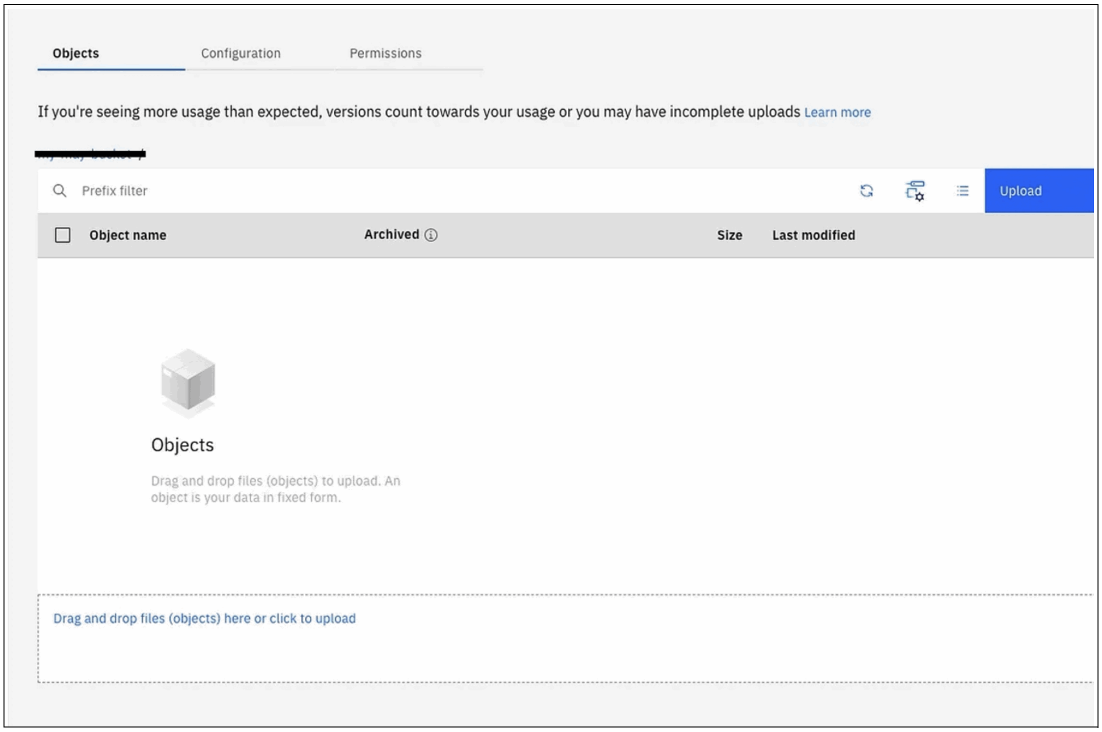


Figure 5-8 Spark engine Start and Stop

9. In the next step, we will associate a bucket with a catalog. We will use an IBM Cloud Object Storage bucket as an example. If you do not already have one, you can create one at this link. We will be using a newly created bucket named "my-bucket." See Figure 5-9 on page 71.



Objects | Configuration | Permissions

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more

[redacted]

Prefix filter

Upload

□	Object name	Archived ⓘ	Size	Last modified

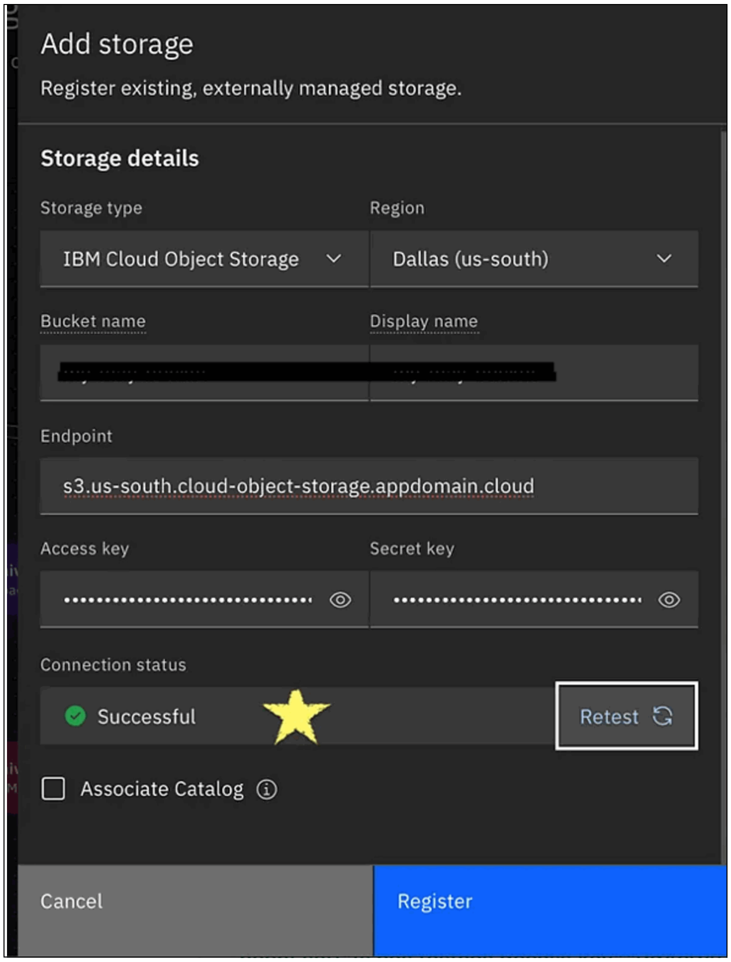
Objects

Drag and drop files (objects) to upload. An object is your data in fixed form.

Drag and drop files (objects) here or click to upload

Figure 5-9 Create an IBM Object Storage instance

10. Next, you need to test the connection. For IBM Cloud Object storage, choose a direct endpoint. In the example shown in Figure 5-10 on page 72, we use a public endpoint.



Add storage

Register existing, externally managed storage.

Storage details

Storage type: IBM Cloud Object Storage

Region: Dallas (us-south)

Bucket name: [redacted]

Display name: [redacted]

Endpoint: s3.us-south.cloud-object-storage.appdomain.cloud

Access key: ••••••••••••••••••••••••••••••

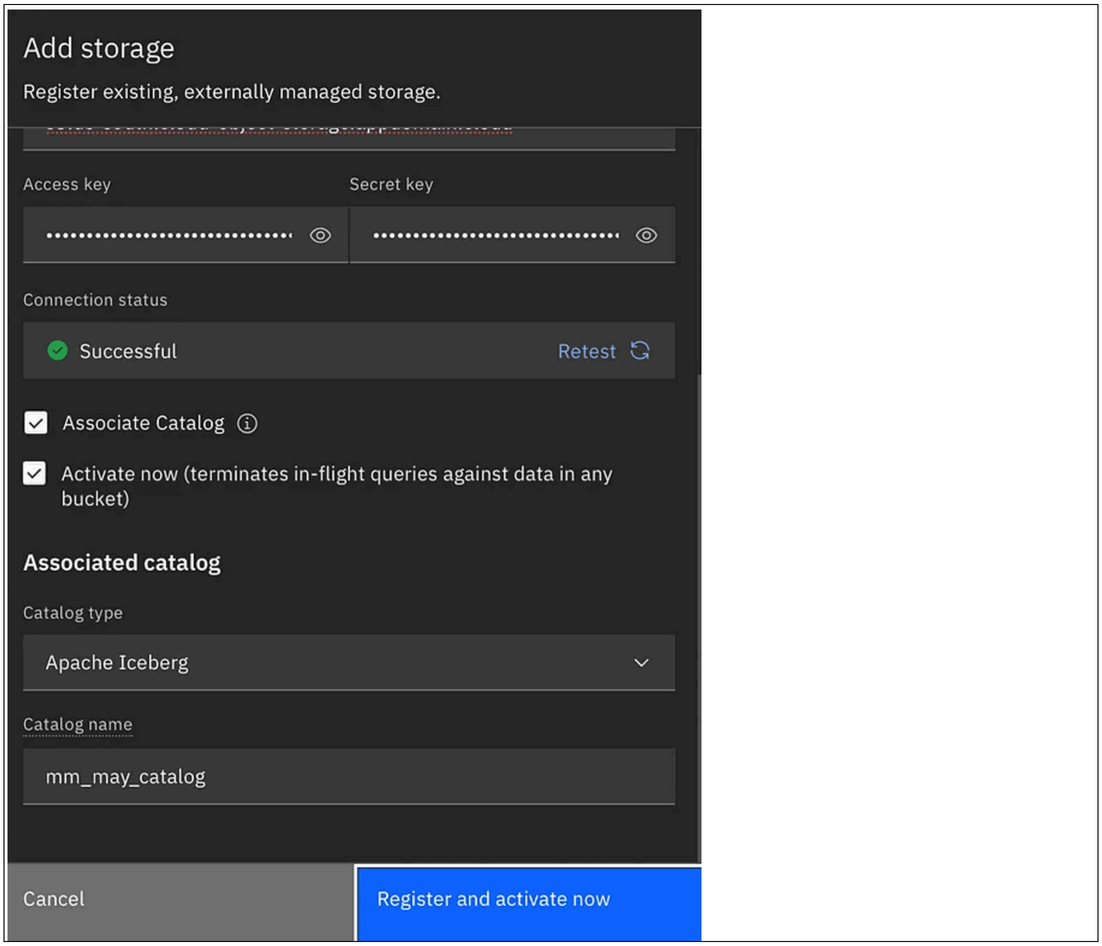
Secret key: ••••••••••••••••••••••••••••••

Connection status: Successful

□ Associate Catalog

Figure 5-10 Add the credentials

11.Next, you add the storage. See Figure 5-11 on page 73.



Add storage

Register existing, externally managed storage.

Access key: ••••••••••••••••••••••••••••••••

Secret key: ••••••••••••••••••••••••••••••••

Connection status: Successful

☑ Associate Catalog

☑ Activate now (terminates in-flight queries against data in any bucket)

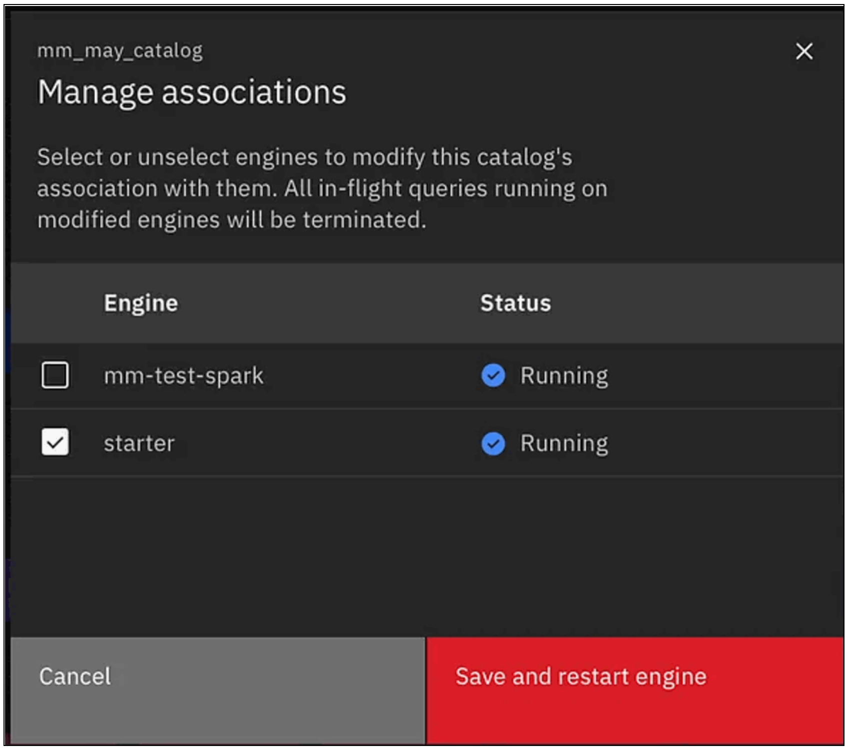
Associated catalog

Catalog type: Apache Iceberg

Catalog name: mm_may_catalog

Figure 5-11 Add the storage

12. Add the associated catalog. This is what was associated with the Native Spark engine in the first section of this chapter. See Figure 5-12 on page 74.


mm_may_catalog

Manage associations

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

	Engine	Status
□	mm-test-spark	✓ Running
☑	starter	✓ Running

Figure 5-12 Manage associations

5.2.2 Explore the tabs in the Spark engine

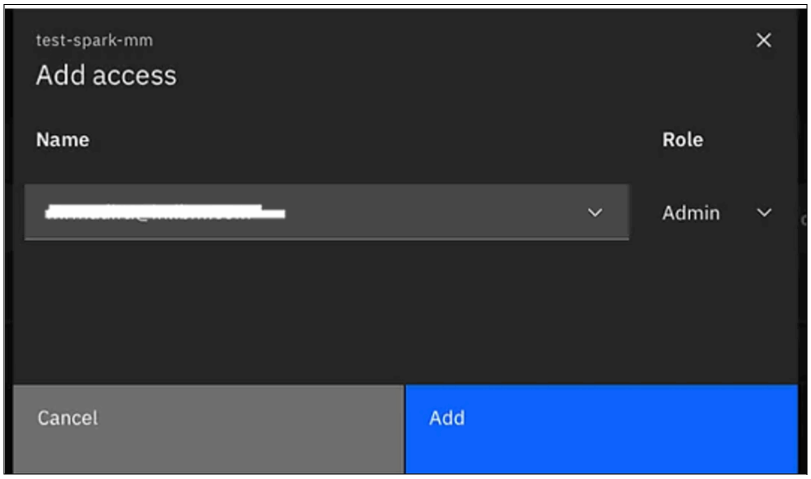
We will explore the following tabs in the Spark engine:

Details tab

The Details tab may seem complex, but it is pre-configured with many settings to simplify setup. You will still need to specify a few essential configurations for the metastore, which we'll discuss later.

Access Control tab

You can invite your team members to the IBM Cloud account and grant admin or user access to the instance (See Figure 5-13 on page 75). To see what privileges you want to grant them, refer to this link.


test-spark-mm

Add access

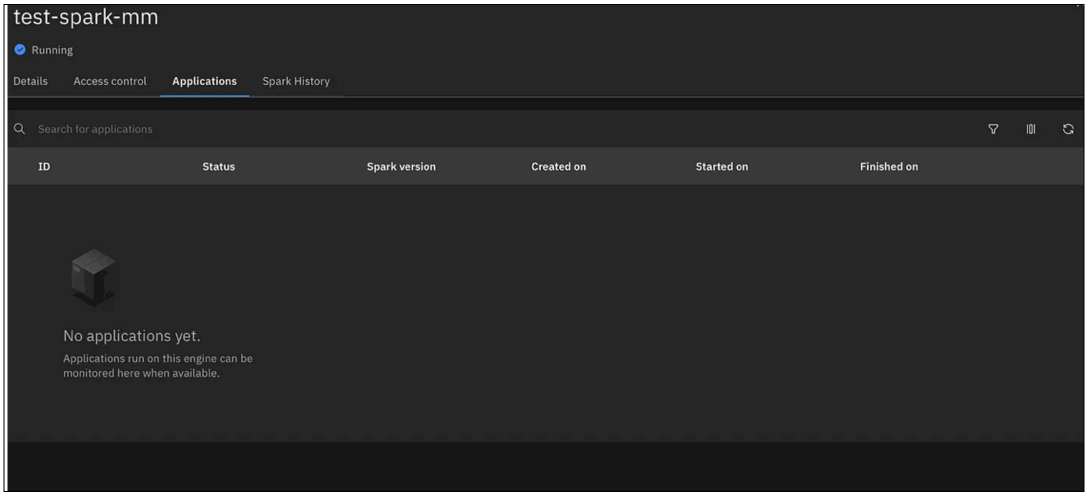
Name: [redacted]

Role: Admin

Figure 5-13   Access control at Spark engine

Applications tab

This section will initially appear empty as no applications have yet been submitted. See Figure 5-14.


test-spark-mm

Running

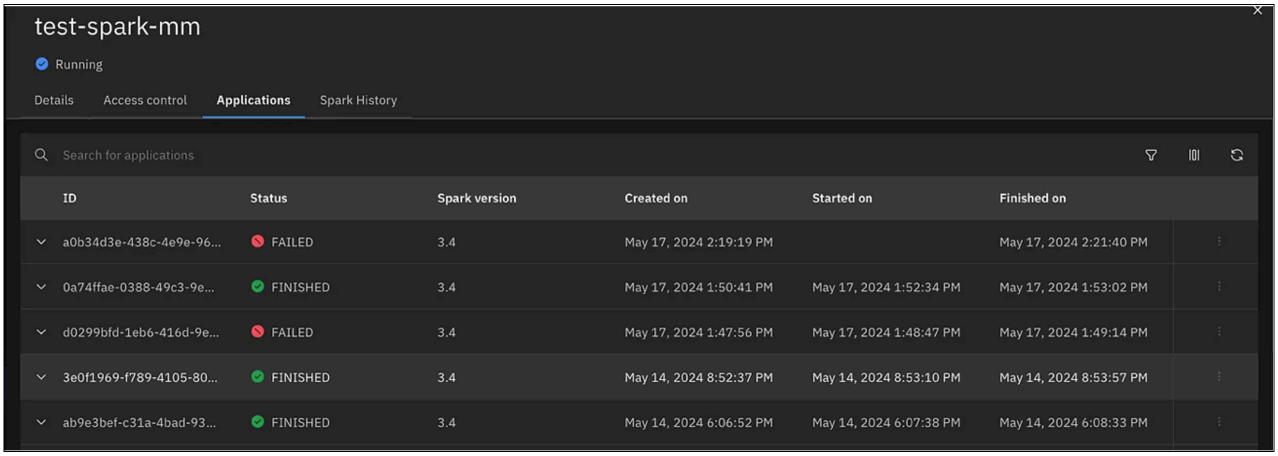
ID	Status	Spark version	Created on	Started on	Finished on

No applications yet.

Applications run on this engine can be monitored here when available.

Figure 5-14   Applications tab

Upon submission of applications, additional details such as status, runtime, and timestamps will be displayed. See Figure 5-15 on page 76.



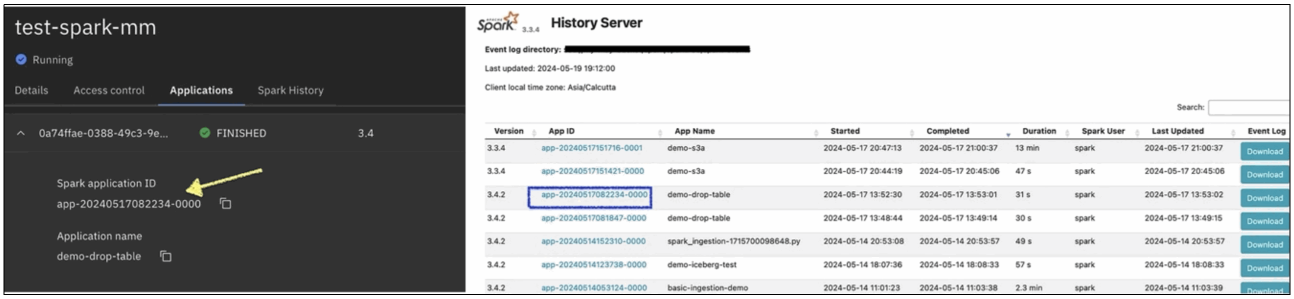
test-spark-mm

Running

ID	Status	Spark version	Created on	Started on	Finished on
a0b34d3e-438c-4e9e-96...	FAILED	3.4	May 17, 2024 2:19:19 PM		May 17, 2024 2:21:40 PM
0a74ffae-0388-49c3-9e...	FINISHED	3.4	May 17, 2024 1:50:41 PM	May 17, 2024 1:52:34 PM	May 17, 2024 1:53:02 PM
d0299bfd-1eb6-416d-9e...	FAILED	3.4	May 17, 2024 1:47:56 PM	May 17, 2024 1:48:47 PM	May 17, 2024 1:49:14 PM
3e0f1969-f789-4105-80...	FINISHED	3.4	May 14, 2024 8:52:37 PM	May 14, 2024 8:53:10 PM	May 14, 2024 8:53:57 PM
ab9e3bef-c31a-4bad-93...	FINISHED	3.4	May 14, 2024 6:06:52 PM	May 14, 2024 6:07:38 PM	May 14, 2024 6:08:33 PM

Figure 5-15 Application status tab

> **Tip:** To locate a particular application within Spark History, utilize its distinctive Spark-generated application ID, as indicated in Figure 5-16.



test-spark-mm

Running

0a74ffae-0388-49c3-9e... FINISHED 3.4

Spark application ID
app-20240517082234-0000

Application name
demo-drop-table

History Server 3.3.4

Event log directory: [redacted]

Last updated: 2024-05-19 19:12:00

Client local time zone: Asia/Calcutta

Search:

Version	App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.3.4	app-20240517151716-0001	demo-s3a	2024-05-17 20:47:13	2024-05-17 21:00:37	13 min	spark	2024-05-17 21:00:37	Download
3.3.4	app-20240517151421-0000	demo-s3a	2024-05-17 20:44:19	2024-05-17 20:45:06	47 s	spark	2024-05-17 20:45:06	Download
3.4.2	app-20240517082234-0000	demo-drop-table	2024-05-17 13:52:30	2024-05-17 13:53:01	31 s	spark	2024-05-17 13:53:02	Download
3.4.2	app-20240517081847-0000	demo-drop-table	2024-05-17 13:48:44	2024-05-17 13:49:14	30 s	spark	2024-05-17 13:49:15	Download
3.4.2	app-20240514152310-0000	spark_ingestion-1715700098648.py	2024-05-14 20:53:08	2024-05-14 20:53:57	49 s	spark	2024-05-14 20:53:57	Download
3.4.2	app-20240514123738-0000	demo-iceberg-test	2024-05-14 18:07:36	2024-05-14 18:08:33	57 s	spark	2024-05-14 18:08:33	Download
3.4.2	app-20240514053124-0000	basic-ingestion-demo	2024-05-14 11:01:23	2024-05-14 11:03:38	2.3 min	spark	2024-05-14 11:03:39	Download

Figure 5-16 Spark History Server

You can monitor and analyze running applications using the Spark UI, which offers a similar interface to Spark History. Click on an application ID in the **Applications tab** to access detailed information about that specific application.

5.2.3 Submitting the application to Native Spark engine

Here we demonstrate how to submit applications using the REST API.

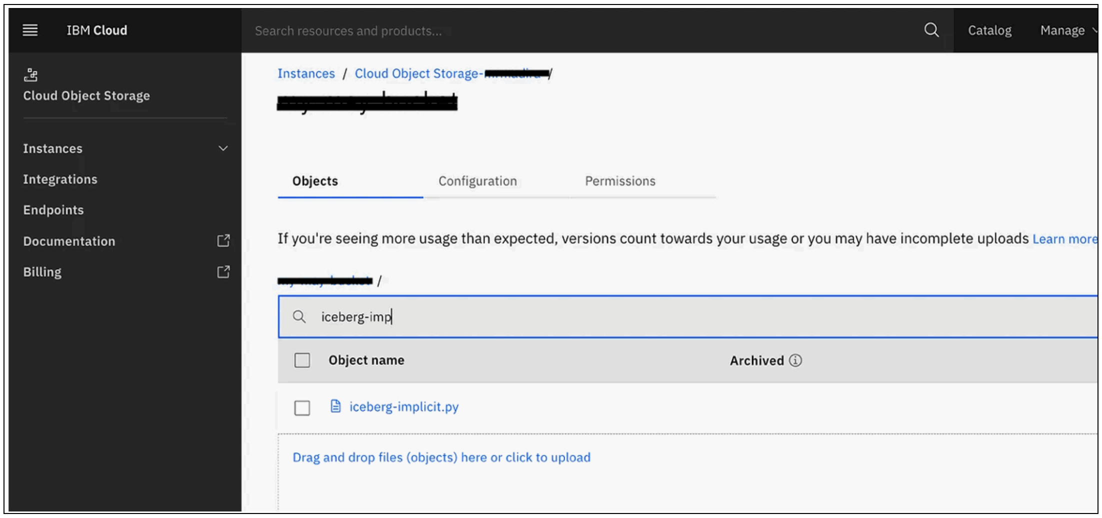
Spark application details

This scenario demonstrates a basic application that creates a database, establishes an Iceberg table, populates it with data, and then retrieves data from the same table. Example 5-1 shows the REST API usage for this scenario.

Example 5-1 REST API usage on Notebook

```
from pyspark.sql import SparkSession
def init_spark():
  spark = 
SparkSession.builder.appName("demo-iceberg-test").enableHiveSupport().getOrCreate(
)
  sc = spark.sparkContext
  return spark,sc
def main():
  spark,sc = init_spark()
  spark.sql("create database if not exists mm_may_catalog.mayday_db1 LOCATION
's3a://my-bucket/'")
  spark.sql("show databases from my_catalog").show()
  spark.sql("create table if not exists my_catalog.db1.testTable1(id INTEGER, name
VARCHAR(10), age INTEGER, salary DECIMAL(10, 2)) using iceberg").show()
  spark.sql("insert into my_catalog.db1.testTable1 values(4,'John
black',23,3400.00),(5,'Peter black',30,5500.00),(6,'George Black',35,6500.00)")
  spark.sql("select * from my_catalog.db1.testTable1").show()
if __name__ == '__main__':
  main()
```

1. First, upload Spark application to a bucket. See Figure 5-17.



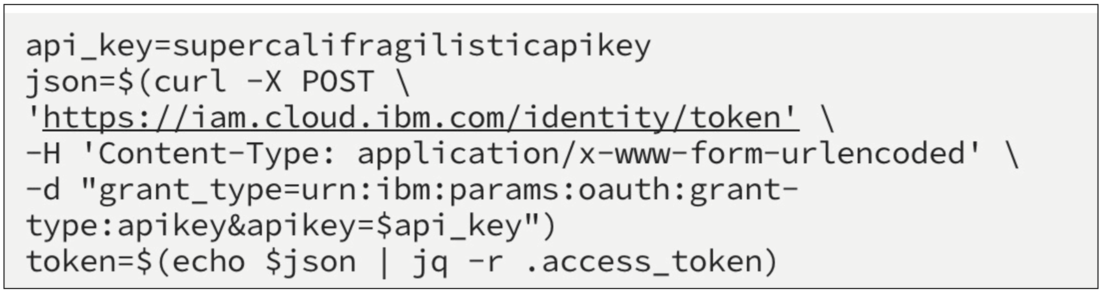
If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads

iceberg-imp

Object name	Archived
iceberg-implicit.py	

Figure 5-17 Upload Spark application to a bucket

2. To submit applications you need a token. To generate the token, you can use the IBM Cloud IAM CLI or use the REST API as shown in Figure 5-18.



```
api_key=supercalifragilisticapikey
json=$(curl -X POST \
'https://iam.cloud.ibm.com/identity/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-d "grant_type=urn:ibm:params:oauth:grant-
type:apikey&apikey=$api_key")
token=$(echo $json | jq -r .access_token)
```

Figure 5-18 Generate an IAM token

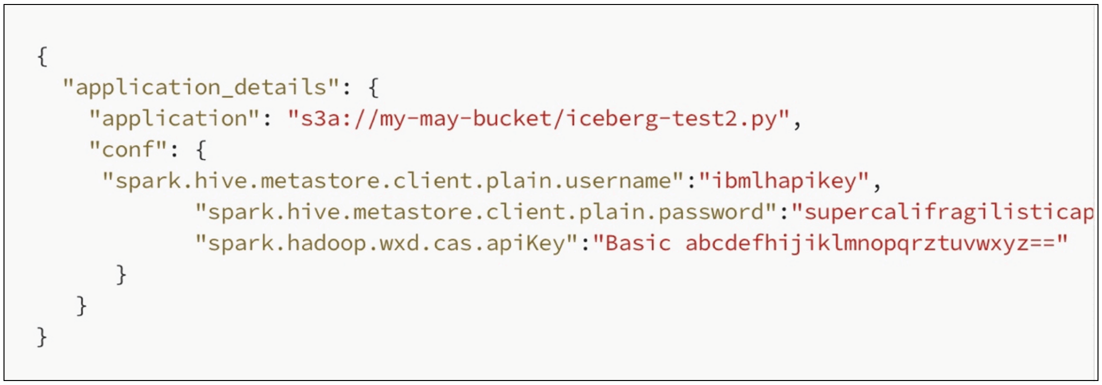
3. Example 5-2 shows the REST API to submit application.

Example 5-2 REST API to submit application

```
instance_id=spark788
crn=crn:v1:bluemix:public:lakehouse:us-south:a/3422342342e:3232-nsd-a83s-abcd-sdf7
sadf::
```

```
curl -v  -X POST
https://us-south.lakehouse.cloud.ibm.com/lakehouse/api/v2/spark_engines/$instance_
id/applications -H "AuthInstanceID: $crn"  --header "Authorization: Bearer $token"
-H "content-type: application/json"  -d @submit.json
```

4. In the curl (Example 5-2 on page 77), the submit.json application payload is shown in Figure 5-19.

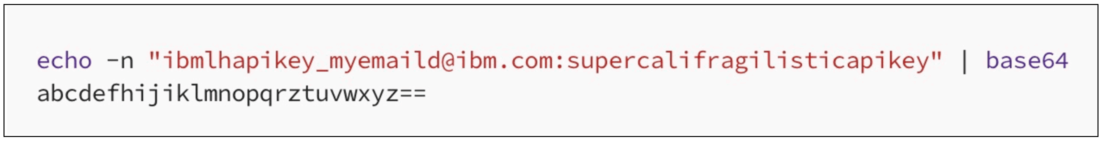


```
{
  "application_details": {
    "application": "s3a://my-may-bucket/iceberg-test2.py",
    "conf": {
     "spark.hive.metastore.client.plain.username":"ibmlhapikey",
            "spark.hive.metastore.client.plain.password":"supercalifragilisticap
            "spark.hadoop.wxd.cas.apiKey":"Basic abcdefhijiklmnopqrztuvwxyz=="
       }
    }
}
```

Figure 5-19   Application payload

Tip: The value for spark.hadoop.wxd.cas.apiKey should be in the format `Basic base64`.

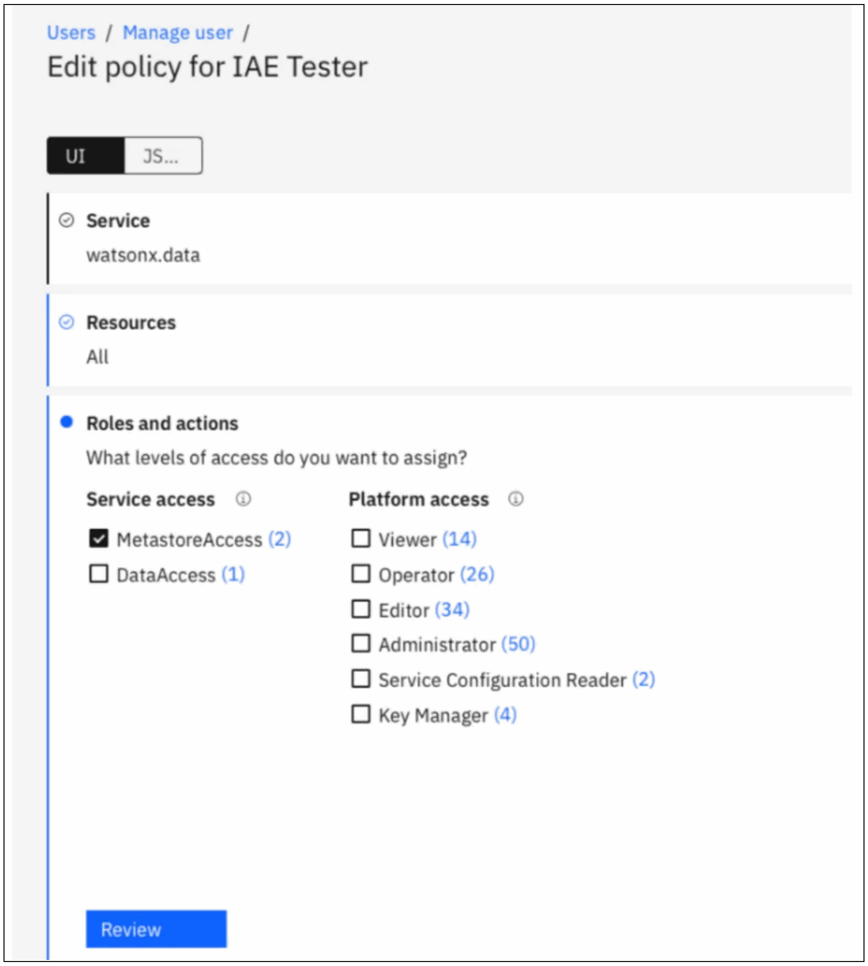
For example, we generated the value as shown in Figure 5-20 on page 78. Replace myemailid@ibm.com with your id. And replace the password with your own apikeyibmlhapikey_ibmcloudid:apikey).



```
echo -n "ibmlhapikey_myemaild@ibm.com:supercalifragilisticapikey" | base64
abcdefhijiklmnopqrztuvwxyz==
```

Figure 5-20   Generate value for specific acccount

Tip: The user submitting the application needs to have MetastoreAccess permission for Spark applications that need to use metastore. Use IAM policies to add the access as shown in Figure 5-21.



Users / Manage user /

Edit policy for IAE Tester

Service

watsonx.data

Resources

All

Roles and actions

What levels of access do you want to assign?

Service access

☑ MetastoreAccess (2)
□ DataAccess (1)

Platform access

□ Viewer (14)
□ Operator (26)
□ Editor (34)
□ Administrator (50)
□ Service Configuration Reader (2)
□ Key Manager (4)

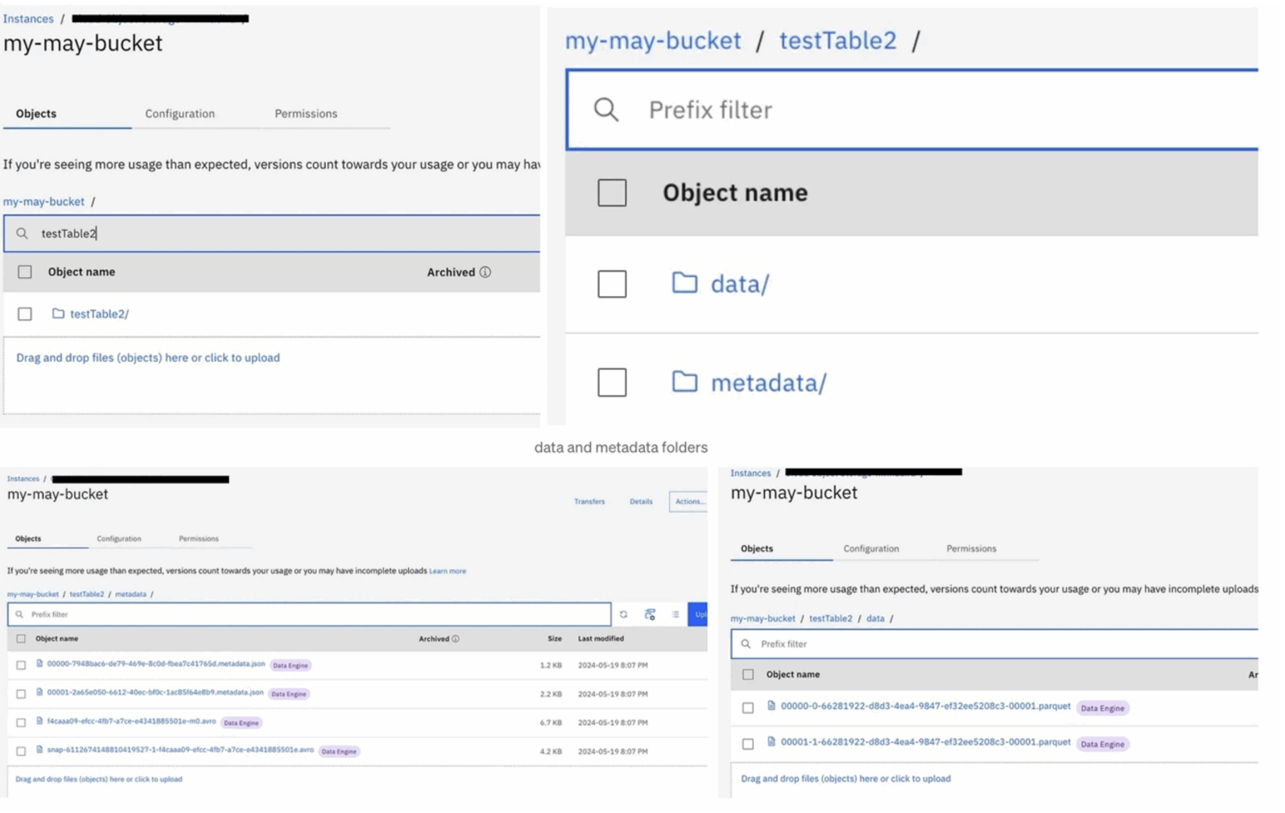
Figure 5-21 Access policy edit

5. Once you submit the application, you get an id and an initial state. You can switch to **Applications** tab of the Spark engine or use another API to track the state of the application.

   You also need this Application ID to debug and search for the logs, as we will show later.

   ```
   {"id":"8ac12670-8fef-4c67-b2ea-b7bd0d5529f2","state":"accepted"}
   ```

6. You can confirm that you can iceberg "style" of data and metadata folders got created.



Instances / [redacted]

my-may-bucket

If you're seeing more usage than expected, versions count towards your usage or you may hav

my-may-bucket /

testTable2

□ Object name	Archived ⓘ
□ testTable2/	

Drag and drop files (objects) here or click to upload

my-may-bucket / testTable2 /

□ Object name
□ data/
□ metadata/

data and metadata folders

Instances / [redacted]

my-may-bucket

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more

my-may-bucket / testTable2 / metadata /

□ Object name	Archived ⓘ	Size	Last modified
□ 00000-7948bac6-de79-469e-8c0d-fbea7c41765d.metadata.json Data Engine		1.2 KB	2024-05-19 8:07 PM
□ 00001-2a65e050-6612-40ec-bf0c-1ac85f64e8b9.metadata.json Data Engine		2.2 KB	2024-05-19 8:07 PM
□ f4caaa09-efcc-4fb7-a7ce-e4341885501e-m0.avro Data Engine		6.7 KB	2024-05-19 8:07 PM
□ snap-6112674148810419527-1-f4caaa09-efcc-4fb7-a7ce-e4341885501e.avro Data Engine		4.2 KB	2024-05-19 8:07 PM

Drag and drop files (objects) here or click to upload

Instances / [redacted]

my-may-bucket

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads

my-may-bucket / testTable2 / data /

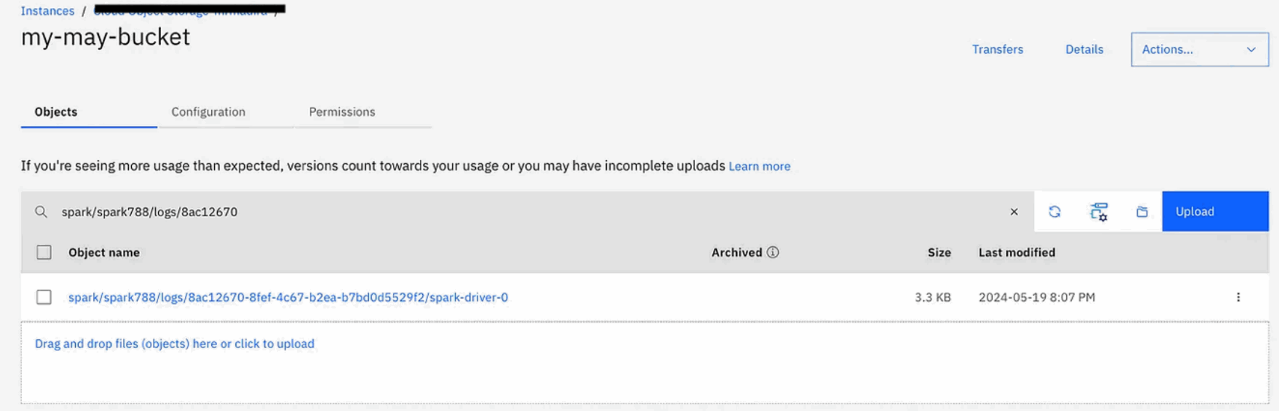
□ Object name	Ar
□ 00000-0-66281922-d8d3-4ea4-9847-ef32ee5208c3-00001.parquet Data Engine	
□ 00001-1-66281922-d8d3-4ea4-9847-ef32ee5208c3-00001.parquet Data Engine	

Drag and drop files (objects) here or click to upload

Figure 5-22 Iceberg data in Object Storage

7. For any Spark application, you may need to examine logs, including those from your application code (for example, show(), print()), as well as the Spark executor and driver logs. For more information, see Debug the Spark application.

> **Tip:** You can type the following in the filter box of the home instance bucket of your Spark engine spark/spark788/logs/8ac12670, which is of the format spark/<engineID>/logs/<first-few-characters-instance-crn>. See Figure 5-23.



Instances / [redacted]

my-may-bucket

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more

spark/spark788/logs/8ac12670

□ Object name	Archived ⓘ	Size	Last modified
□ spark/spark788/logs/8ac12670-8fef-4c67-b2ea-b7bd0d5529f2/spark-driver-0		3.3 KB	2024-05-19 8:07 PM

Drag and drop files (objects) here or click to upload

Figure 5-23 Application logs

5.3 Execute important queries using the power of traditional RDBMS with shared open lakehouse formats

As organizations increasingly embrace the flexibility and scalability of lakehouse architectures - fusing the schema-on-read philosophy of data lakes with the performance characteristics of data warehouses - one challenge often emerges: how to preserve the established benefits of traditional relational systems. Longstanding RDBMS platforms have earned their place in enterprise environments by providing unparalleled transactional consistency, robust optimization, and universally understood access methods via SQL. Rather than discarding these hard-won strengths, watsonx.data intelligently integrates them into the evolving data ecosystem, ensuring that modern data architectures gain the reliability, performance, and user-friendliness traditionally associated with relational databases.

This integration is not about forcing a strict schema on all data sources, nor about replicating legacy infrastructure within a new paradigm. Instead, watsonx.data applies the core principles and best practices honed over decades of RDBMS evolution and extends them to open, columnar file formats, distributed storage environments, and hybrid cloud infrastructures. The result is a solution that provides end-to-end trustworthiness from ingestion to query execution without sacrificing openness or scalability.

5.3.1 ACID guarantees transactional reliability

In data-intensive enterprises, the importance of data correctness cannot be overstated. Transactional reliability underpins critical operations, from processing financial trades to updating inventory counts and handling personal health information. A single corrupted record or partial update can ripple through downstream analytics, misinform decision-makers, violate compliance standards, or damage the organization's reputation.

Relational databases have long offered robust transactional support by adhering to ACID principles, as defined below:

- Atomicity: Each transaction is treated as an indivisible unit; it either completes fully or not at all.
- Consistency: All data written adheres to predefined rules, maintaining referential integrity and validity.
- Isolation: Transactions run concurrently without interfering with each other's intermediate states.
- Durability: Completed transactions are permanently recorded, ensuring that committed changes persist even after system failures.

These properties created a trustworthy environment for mission-critical systems and provided a bedrock of reliability that organizations came to rely on.

IBM watsonx.data carries these transactional semantics into environments where data may reside in a combination of object stores, on-premises file systems, or distributed cloud storage-often using open file formats like Parquet, ORC, or Delta Lake. Implementing ACID in such a heterogeneous ecosystem involves careful orchestration of metadata operations, versioning, and concurrency control across large, possibly partitioned datasets.

The platform ensures that data transformations, such as batch ingestion, schema evolutions, incremental updates, or data masking, happen in an all-or-nothing manner, just as they would in a traditional RDBMS. Thus, watsonx.data preserves the consistency and trustworthiness of data, even as it spans multiple file-based storage layers. Organizations gain the confidence that their analytics and reporting pipelines are grounded in a stable and reliable foundation, reducing the risk of erroneous insights and regulatory non-compliance.

5.3.2 Advanced query optimization

Optimizing queries is as much an art as it is a science in the database world. Traditional RDBMS platforms have refined their optimization engines over decades, employing a deep understanding of data distribution, indexing structures, join algorithms, and CPU and memory utilization patterns. These optimizations allow complex analytical workloads-like intricate joins between large tables, multi-level aggregations, and subqueries to execute within practical time frames.

RDBMS query optimizers rely on comprehensive metadata and statistics about the underlying data. They analyze the sizes of tables, distribution of values, existing indexes, and cost models to determine the most efficient execution plan. Whether scanning a massive fact table, leveraging a sorted index, or pushing filters down closer to the data, these systems have become adept at minimizing resource consumption and delivering interactive response times for queries that would otherwise be unwieldy.

Migrating these capabilities into a lakehouse context requires extending the optimization logic beyond traditional row-store tables and B-trees. IBM watsonx.data works with columnar formats like Parquet, ORC, and Delta, which store data in a way that naturally supports predicate push-down, column pruning, and partition elimination. The system leverages metadata embedded within these formats, including min/max statistics per column and file partitions, to minimize I/O overhead and reduce the query's data footprint.

Additionally, watsonx.data's optimizer is designed to understand the cost of operations in distributed environments, where network traffic and storage access patterns matter as much as CPU usage. By dynamically evaluating which subsets of data are relevant and which operations can be parallelized or pruned, the engine can execute complex queries against immense datasets with surprising speed. This empowers analysts to explore data interactively, iterate on hypotheses, and refine models without suffering from long turnaround times or hefty data movement costs.

5.3.3 Standard SQL support

Despite the proliferation of new programming languages, frameworks, and query paradigms, SQL endures as the default language of choice for analytics. Its declarative syntax, standardized grammar, and broad support across numerous tools and platforms make it accessible to a vast audience-business analysts, data scientists, developers, and even managers with technical acumen. SQL's universality and expressiveness have kept it at the center of data querying and manipulation.

In classical relational settings, SQL acts as a powerful yet relatively simple interface that abstracts away the complexity of underlying data structures. Users do not need to write intricate, low-level code to retrieve data or perform computations; they simply specify what they want, and the system figures out how to deliver it efficiently. This abstraction layer fuels productivity, reduces the learning curve, and fosters a self-service culture where more people can interact directly with organizational data.

The challenge in modern lakehouse environments is that data is no longer confined to well-defined relational schemas. Instead, it may exist as nested JSON fields, denormalized tables, or raw files sprinkled across various storage layers. IBM watsonx.data bridges this gap by allowing users to treat these open-format files as if they were relational tables. A virtualized metadata layer provides logical schemas for these files, allowing familiar SQL constructs like SELECT, JOIN, GROUP BY, and WINDOW functions to operate seamlessly across them.

By providing a uniform SQL layer over both structured warehouse data and semi-structured lakehouse data, watsonx.data drastically simplifies the analytical workflow. Teams do not need to master new query languages or develop custom parsers to access and manipulate data in Parquet or ORC files. Instead, they can rely on their existing SQL knowledge, reusing queries, tools, and dashboards that were originally designed for relational systems. This not only boosts productivity but also broadens the user base that can leverage the data ecosystem, encouraging data-driven insights to permeate the entire organization.

5.3.4 Embracing lakehouse architecture and open formats

As enterprises strive to unify their data strategies, the lakehouse architecture has emerged as a key paradigm that combines the strengths of traditional data warehouses-such as structured schemas, performance optimizations, and reliability-with the flexibility, scalability, and cost efficiencies characteristic of data lakes. This hybrid approach addresses longstanding tensions in the data landscape, allowing organizations to benefit from the openness and extensibility of data lakes without sacrificing the governance, consistent performance, and analytical sophistication of a warehouse.

In practice, embracing the lakehouse model means handling data in a way that is format-agnostic, interoperable across multiple platforms, and conducive to evolving analytical requirements. IBM watsonx.data operationalizes these principles by providing a platform that not only leverages open, widely adopted storage formats and metadata frameworks but also integrates governance and performance enhancements. The result is a single, cohesive data environment that adapts to changing workloads, heterogeneous data types, and evolving regulatory landscapes-all while maintaining the transactional guarantees and robust analytics capabilities that enterprises have come to rely on.

Open-source storage formats

A pivotal aspect of the lakehouse paradigm is the use of open, standardized storage formats. Gone are the days when organizations were locked into proprietary schemas or tied to a single vendor's file formats. Instead, open columnar formats like Parquet, ORC, and open table formats such as Delta Lake have become industry-standard choices for storing large analytical datasets. IBM watsonx.data's embrace of these formats unlocks a range of benefits, including:

- Efficient compression and encoding: High-performance columnar formats like Parquet and ORC are designed to store data by columns rather than by rows. This structure allows for more effective compression, especially if columns exhibit low cardinality or repetitive patterns. Consequently, organizations can drastically reduce storage costs while simultaneously lowering I/O overhead. In cloud environments, this translates directly to cost efficiencies, as less data is read and transferred over the network.
- Schema evolution: Rigid, pre-defined schemas can stifle innovation and complicate data integration. Open formats support schema-on-read and schema evolution, enabling organizations to gradually adjust table structures as new attributes emerge, old attributes become obsolete, or analytical needs evolve. This flexibility reduces the need for costly and time-consuming migrations or reingestion processes.
- Predicate push-down and data pruning: Intelligent push-down of filters and pruning of unnecessary data segments allow query engines to skip irrelevant files and columns. By filtering data at the source, systems can minimize the amount of data scanned and returned during queries, delivering faster results and more efficient use of compute resources.

By natively supporting these open formats, watsonx.data ensures that enterprises are not dependent on vendor-specific platforms or confined to a single ecosystem. This vendor-neutral stance not only future-proofs data investments but also enables businesses to mix and match best-in-class tools-ranging from data science notebooks and BI dashboards to machine learning frameworks and data cataloging services without introducing compatibility roadblocks. Over time, organizations can seamlessly adopt emerging technologies, move workloads between clouds, and integrate with evolving data stacks while retaining the storage foundation that open formats provide.

Metadata management and governance

As data environments grow in scale and complexity, a critical challenge emerges - ensuring that analysts, engineers, and data stewards can find, understand, trust, and appropriately secure the data they need. Without robust metadata management and governance frameworks, even the most flexible lakehouse can become difficult to navigate, resulting in *data swamps* where valuable insights are submerged beneath an ocean of unstructured files.

IBM watsonx.data addresses these challenges by providing a comprehensive metadata layer that elevates raw files into discoverable, well-defined, and governable entities. Key capabilities include:

- Centralized data catalogs and discovery: Analysts can search and browse data catalogs to quickly locate relevant datasets. Metadata attributes, including schema definitions, data lineage, data quality scores, business glossary terms, and data owners, help users identify the most suitable data for their analyses. This fosters a culture of self-service analytics, where teams can find what they need without constantly relying on IT gatekeepers.
- Consistent schema definitions and versioning: By imposing logical schemas atop raw files, watsonx.data ensures that all users refer to the same, consistent representation of data. Centralized schema repositories maintain historical versions, enabling analysts to *time travel* to older schemas if needed. This reduces confusion, accelerates onboarding for new team members, and supports stable reporting over long periods.
- Data lineage and provenance tracking: Complex analytical pipelines often involve multiple transformations, joins, and enrichments. Without lineage tracking, it can be nearly impossible to reconstruct how a particular metric was derived or to assess the impact of a schema change on downstream analytics. IBM watsonx.data's metadata layer captures lineage information, providing transparency and auditability. This not only aids troubleshooting and regression analysis but also strengthens compliance efforts by documenting data transformations and validating data sources.
- Security, compliance, and access controls: Regulatory mandates like GDPR, HIPAA, or CCPA often require stringent controls over who can see which data and under what conditions. IBM watsonx.data integrates with enterprise security frameworks, enabling role-based access control, encryption at rest and in transit, and data masking. Sensitive information-like personal identifiers-can be masked or tokenized to protect privacy. These governance measures ensure that analytics remain safe, compliant, and trustworthy, while still allowing analysts to work effectively with the data.

Performance enhancements for analytical queries

A central tenet of the lakehouse philosophy is delivering data warehouse-like performance on top of flexible, open storage. This balance demands more than just fast hardware: it requires a deep integration between query engines, metadata systems, and the underlying file formats to deliver sub-second or interactive query speeds over massive, ever-growing datasets.

IBM watsonx.data fully embraces the performance-oriented features available in lakehouse architectures, including:

- Partitioning and clustering: By logically grouping related data-such as by date, region, product category, or other attributes-organizations can localize queries to specific partitions, reducing the volume of data scanned. Clustering similar data together further improves efficiency by enhancing data locality and compression ratios.
- Z-ordering and data skipping: Advanced file layout techniques like Z-ordering rearrange data to maximize locality across multiple dimensions. This technique can dramatically reduce the search space for queries filtering on multiple columns. Additionally, file-level statistics and indexes enable skipping entire files that do not match query predicates, further trimming query execution times.
- Indexing and summary statistics: Just like traditional databases, lakehouse systems can maintain indexing structures and summary statistics on column distributions. These help the query optimizer quickly identify which segments of data are most relevant, leading to more efficient scans and joins. Instead of sifting through all data, watsonx.data can pinpoint only the necessary subsets, delivering near-interactive query responses.
- Caching and adaptive execution: Frequently accessed data can be cached in memory or on faster storage tiers to accelerate subsequent queries. Adaptive execution strategies allow watsonx.data to modify query plans on the fly, reacting to real-time performance metrics and data distribution patterns. The system can split large tasks, redistribute workload, or change join strategies mid-query to achieve optimal performance.

By leveraging these performance enhancements, watsonx.data ensures that analytics on even colossal datasets remain responsive and cost-effective. Analysts gain the ability to iterate rapidly over complex queries, data scientists can train machine learning models at scale, and business users enjoy timely insights without waiting hours for batch processing jobs to complete

6

Establishing data governance

IBM watsonx.data offers a comprehensive data governance solution. This powerful platform empowers organizations to ensure data security, compliance, and usability.

In this chapter we discuss the following:

6.1 Governing your data: The role of catalog, metadata, and policies

The concepts of catalog, metadata, and policies are integral to data governance and management within the watsonx platform.

- Catalog: In watsonx, the catalog acts as a centralized hub for managing and organizing data assets. It provides a unified view of all available data, regardless of its source or location. This allows users to easily discover, understand, and access the data they need for their AI and analytics projects. The catalog not only lists data assets but also provides tools for collaboration and knowledge sharing, enabling teams to work together more effectively.
- Metadata: Metadata plays a crucial role in providing context and meaning to the data within the watsonx catalog. It includes information such as data lineage, data quality metrics, business terms, and technical definitions. It is important to implement active metadata management, which involves automatically capturing and updating metadata to ensure its accuracy and completeness. This enables users to understand the characteristics of the data, its origin, and how it can be used, fostering trust and confidence in data-driven insights.

  IBM watsonx.data has a central storage for information about its data, called the Metadata Service. This lets different tools work together seamlessly because they all understand the data the same way.
- Policies: Policies in watsonx define the rules and guidelines for how data can be accessed and used. They help ensure data privacy, security, and compliance with regulatory requirements. It is important to implement granular policy controls, which allow organizations to define policies at various levels, from individual data assets to entire catalogs. This enables organizations to enforce data governance policies consistently and effectively across their data landscape.

Together, the catalog, metadata, and policies in watsonx provide a comprehensive framework for governing and managing data assets. They enable organizations to unlock the full potential of their data while ensuring trust, transparency, and compliance.

6.2 Best practices for implementing an effective data governance framework

In this section we discuss some best practices for implementing an effective data governance framework in watsonx.data.

6.2.1 Cataloging

The following are best practices for cataloging:

- Start with a clear data inventory: Identify all your data sources and assets, including structured, unstructured, and semi-structured data. This comprehensive view will help prioritize governance efforts.
- Leverage IBM Knowledge Catalog (IKC) integration: IBM watsonx.data integrates seamlessly with IKC, enabling a centralized catalog across your entire data ecosystem. This promotes consistency and simplifies data discovery. 6.3, “Integration with IBM Knowledge Catalog (IKC)” on page 90 discusses this integration in depth.

- Standardize data organization: Implement a consistent naming convention and classification system for data assets within the catalog. This improves searchability and understanding for all users.
- Utilize business glossaries: Define clear and concise business terms within the catalog to ensure everyone interprets data attributes the same way.

6.2.2 Metadata management

The following are best practices for metadata management:

- Adopt a metadata governance strategy: Define clear ownership and responsibilities for capturing, validating, and maintaining metadata. This ensures its accuracy and consistency.
- Automate metadata capture: Utilize watsonx.data's automated metadata extraction capabilities to minimize manual efforts and reduce errors.
- Enrich metadata with business context: Go beyond technical data definitions. Include lineage information, quality metrics, and usage guidelines to provide a richer understanding of the data.
- Integrate metadata with AI workflows: Use high-quality metadata to improve the accuracy and effectiveness of your AI models and analytics initiatives.

6.2.3 Policy management

The following are best practices for policy management:

- Align policies with business objectives: Develop data governance policies that support your organization's goals and regulatory requirements.
- Implement role-based access control (RBAC): Define clear access levels and permissions for users based on their roles and responsibilities. This ensures data security and minimizes the risk of unauthorized access.
- Leverage data usage tracking: Utilize IBM watsonx.data's data lineage and usage tracking features to monitor how data is being accessed and used. This helps identify potential compliance issues and promotes responsible data practices.
- Promote data governance awareness: Educate and train users on your data governance policies and procedures. This fosters a culture of data responsibility within your organization.

6.2.4 General best practices

It is important to start small and scale gradually. You can begin by implementing data governance on a pilot project and gradually expand it across your organization. This allows for adjustments and refinements based on user feedback.

Involve key stakeholders from different departments in the data governance process. Their input helps ensure that the framework meets the needs of all users.

Regularly assess the effectiveness of your data governance framework and make adjustments as needed. This ensures it remains aligned with your evolving data landscape and business needs.

6.3 Integration with IBM Knowledge Catalog (IKC)

The integration between IBM watsonx.data and IBM Knowledge Catalog enables comprehensive data governance through several aspects such as Data linage, Semantic enrichment, and Data Privacy protection, etc. In this chapter, we focus on how IKC's data protection rules play a critical role in ensuring data privacy as part of the broader data governance framework.

6.3.1 Architecture and core components of the integration

IBM Knowledge Catalog servers as an enterprise data governance solution, managing the metadata for data sources across the enterprise. IBM watsonx.data acts as one of the data sources to support deep policy enforcement - dynamically enforce data protection whenever user access data. Figure 6-1 shows the architecture of the IBM Knowledge Catalog integration.

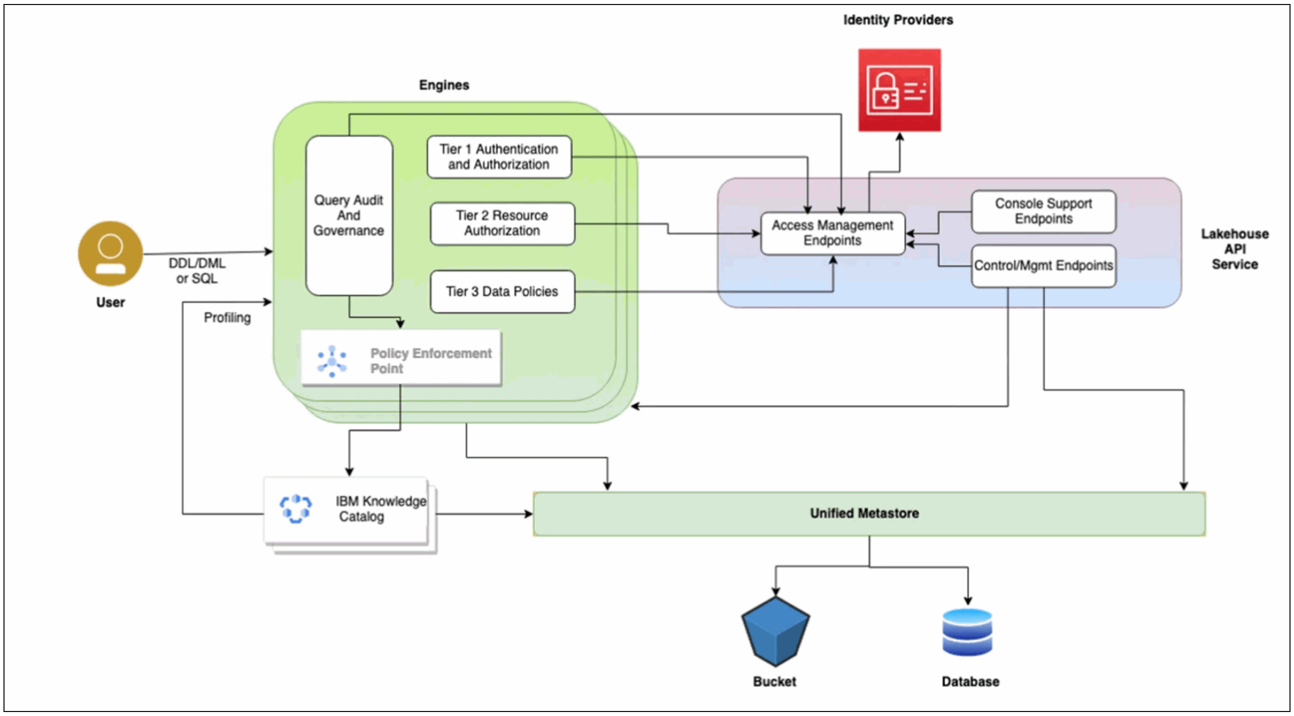


Figure 6-1 Architecture of the integration

In the architecture diagram above, there are several key components involved in the integration.

- Access Management Endpoints: This component is used to facilitate IKC access to the metadata and data of watsonx.data and allow the service token of IKC to bypass the built-in access controls, enabling profiling of raw data and identifying the correct data classes for columns.
- Query Audit and Governance: This component is a Presto engine plugin. When users attempt to access data from watsonx.data, it triggers corresponding data governance requests to IKC and protect the sensitive data using data masking or row level filtering based on the predefined data protection rules in IKC.
- Policy Enforcement Point: This component is used to evaluate the IKC data protection rules using IKC SDKs and cache the policy evaluation results for repeating access.

6.3.2 Implementation of the integration

Before data governance can be fully implemented, several preparatory steps must be completed in the IBM Knowledge Catalog. Typically, a role called *Data Steward*, is responsible on overseeing data governance, importing assets and defining policies/rules to ensure privacy protection.

The Data Steward will need to create a project first, establish a connection to watsonx.data, import data assets, and run metadata enrichment jobs to define business terms, ensuring a common understanding of data assets across the organization. Data Stewards will also need to run data profiling to categorize data assets and associate each column of data to different data classes.

The data protection policies and rules can be defined on top of business terms and data classes to protect sensitive data, ensuring compliance with regulatory requirements and organizational policies. Figure 6-2 on page 92 illustrates the general process of this practice.

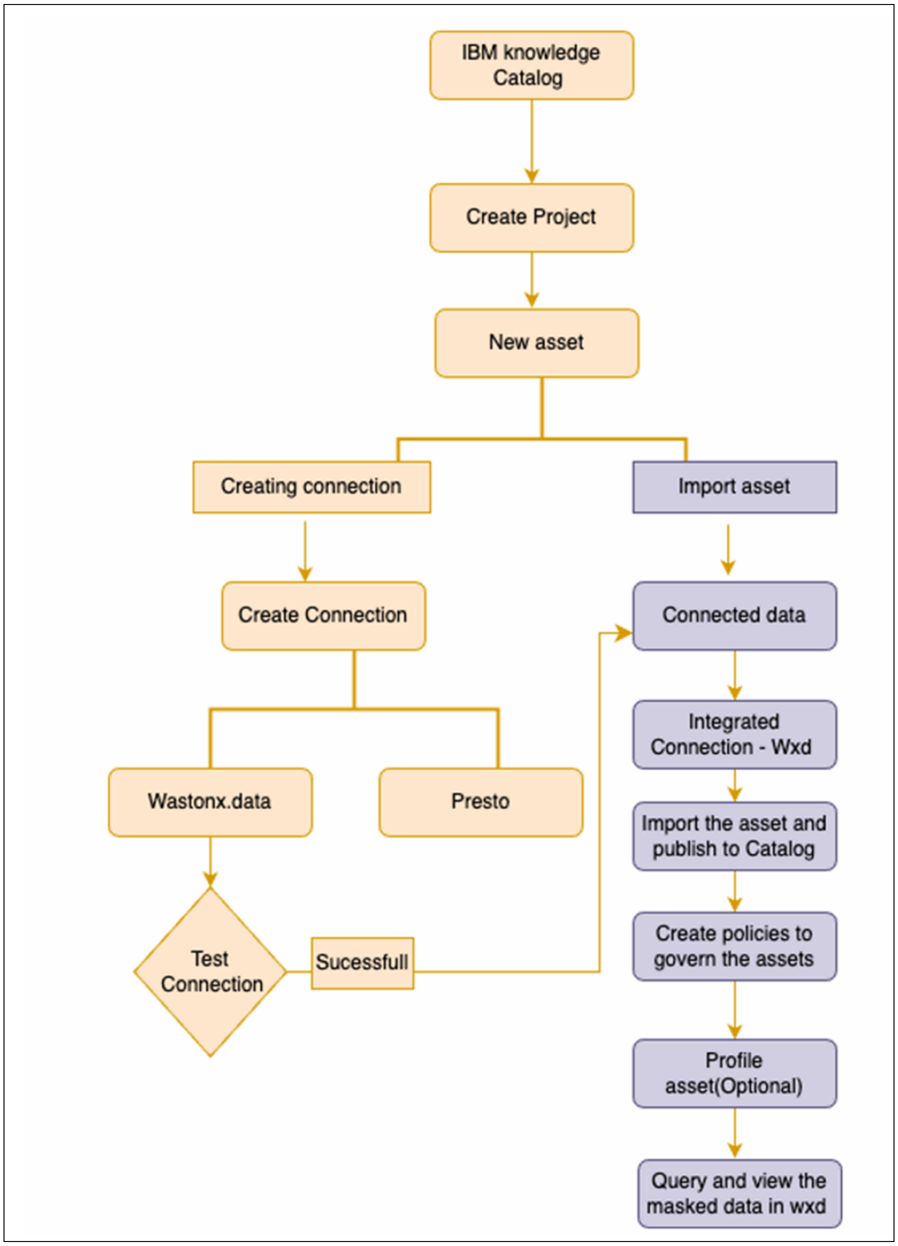


Figure 6-2 The general process to prepare for integration in IKC

Once these preparations are complete, data protection policies and rules are automatically applied when users query data in IBM watsonx.data. For example, if a column contains personally identifiable information (PII) such as Social Security Numbers (SSNs), watsonx.data will automatically transform the data using masking techniques like redaction or obfuscation, complying with data protection rules. This ensures data privacy is maintained during data access.

6.3.3 Summary and references

In summary, the integration of IBM Knowledge Catalog with IBM watsonx.data facilitates the organization and understanding of data, implements privacy data protection such as data redaction, obfuscation, etc. This integration ensures that sensitive data is protected while making data accessible for AI and data-driven initiatives. It provides a strong foundation for enhanced security, compliance, and data governance practices.

For more information, visit IBM watsonx.data documentation.

7

Establishing a data catalog

This chapter describes the steps to establish a data catalog with IBM watsonx.data so that you can do achieve the following goals:

- Unify data sources: Consolidate data from across your organization, regardless of format or location.
- Navigate with ease: Discover relevant data sets quickly by using intuitive search and metadata management.
- Boost data governance: Help ensure data quality and security by using robust access controls and lineage tracking.

This chapter has the following sections:

7.1 Introduction

A data catalog is foundational to any data-driven organization. It acts as a centralized and structured repository for documenting, managing, and categorizing data assets across the enterprise. It is not a passive storage system, but a dynamic resource that facilitates access, discovery, and governance of data. In IBM watsonx.data, the catalog is designed to handle vast amounts of data, and it is a critical component of the organization's data governance strategy. It provides tools for managing metadata to help ensure compliance, enable efficient collaboration, and maintain data security. This catalog is crucial in mitigating data silos and helping ensure that stakeholders, whether technical or business-oriented, have the right tools to access and make sense of the data.

The core function of the IBM watsonx.data catalog is the management and enrichment of metadata, which includes detailed information about the origins, transformations, and lineage of each data asset. This metadata captures technical details, such as data types and formats, and the broader context of how the data was collected, processed, and altered. By using this metadata, users can understand how data flows through the organization, which provides transparency about its lifecycle. Whether the data originates from internal business operations or external sources, IBM watsonx.data helps ensure that every data set is thoroughly documented, which enables the tracing of its lineage to identify potential issues such as data integrity or trustworthiness.

Moreover, the catalog is equipped with robust data governance and security features. It supports role-based access control (RBAC), which enables the organization to enforce permissions that restrict access to sensitive information. For example, personal identifiable information (PII) can be masked or anonymized to meet compliance standards, which help ensure that privacy regulations are respected. Furthermore, IBM watsonx.data promotes collaboration by enabling users to tag data sets with business-context annotations so that others can understand the data sets' relevance in specific use cases. This capability enhances data sharing across departments, which helps stakeholders interpret data more efficiently and makes the catalog more valuable as a comprehensive and centralized resource.

7.2 Data discovery: Automating data classification and tagging for better organization

In IBM watsonx.data, automated data discovery represents a significant innovation that transforms raw, unorganized data sets into structured, valuable assets. Using sophisticated machine learning models, IBM watsonx.data automatically analyzes data based on its attributes and content, and applies classifications and tags that mirror the expertise of human data stewards. This approach eliminates the need for manual tagging, which speeds up the data preparation process and helps ensure that data is always organized and accessible.

The process of data classification within IBM watsonx.data involves advanced entity recognition models, which are tailored to identify relevant entities like dates, locations, or product IDs in text-based data sets. The system also uses *natural language processing (NLP) models* to categorize textual data based on its content. For example, NLP can classify customer feedback as “positive” or “negative” or identify and label entities within a news article. This flexible, multi-model approach makes IBM watsonx.data suitable for a diverse range of data sets, whether structured (for example, relational databases) or unstructured (for example, text documents or multimedia content).

Once these classifications are identified, IBM watsonx.data enables users to define customized tagging rules based on business-specific needs. These rules can consider various data set properties, such as column names, statistical distributions, or data value ranges. The tagging system dynamically applies these rules to incoming data sets, which help ensure that newly ingested data is automatically categorized and tagged according to predefined guidelines. This level of automation increases efficiency and reduces the manual effort that is required to keep the data catalog up to date.

7.3 Data profiling

Data profiling is a key activity that assesses the quality, consistency, and readiness of data sets for analytics. In IBM watsonx.data, profiling is deeply integrated into the data governance process, which enables organizations to monitor and ensure the integrity of their data throughout its lifecycle. Profiling involves the examination of data to determine its quality attributes, such as data types, value distributions, null value percentages, and the presence of outliers. This step is essential for identifying data that might not meet organizational standards or that might distort analysis if used without remediation.

The IBM watsonx.data profiling engine uses advanced statistical methods and machine learning algorithms to rapidly scan large data sets, detect anomalies, and highlight potential data quality issues. For example, the system can flag missing or inconsistent data, or it can identify unexpected patterns that might signify erroneous data entry. In addition to traditional profiling, IBM watsonx.data supports more complex evaluations, such as data relationship analysis and schema validation, which assess how data entities are connected and whether the data schema aligns with expected standards.

Once profiling is complete, IBM watsonx.data integrates the findings into a data quality framework that automatically generates alerts when data sets fall below predefined quality thresholds. These alerts enable data stewards to address issues before they compromise downstream analytics. Also, customizable workflows enable organizations to tailor profiling strategies to meet their unique governance needs, which provide a flexible solution for diverse data environments.

7.4 Data cataloging: Building a comprehensive data catalog for findability

The creation of a comprehensive data catalog in IBM watsonx.data is a multi-step process that involves careful planning, taxonomy development, and strategic organization of data sets. The catalog’s architecture is designed to help ensure that data is *discoverable* and *actionable*. A key aspect is creating a clear *taxonomy* that aligns with organizational goals and the way that data is used across the business. This taxonomy defines the logical categories in which data sets are organized, which creates a hierarchical structure that enables users to quickly find and explore related data sets.

To ensure that the catalog is dynamic and adaptable, IBM watsonx.data provides tools to define metadata standards that specify which attributes should be associated with each data set, such as schema definitions, data sensitivity, update frequency, and ownership. Also, IBM watsonx.data supports the automation of catalog updates through scripts and APIs, which reduce manual overhead and helps ensure that the catalog remains consistent as data assets evolve. With this level of automation, organizations can keep their catalogs up to date without the need for constant human intervention.

IBM watsonx.data also offers *advanced indexing capabilities*, which optimize the catalog for fast and efficient search queries. By indexing the metadata, tags, synonyms, and multi-language support, IBM watsonx.data enhances search performance so that users can search by using various metadata attributes. This flexibility in indexing and search capabilities helps ensure that users can quickly find the data sets that they need, regardless of language, data type, or data set complexity.

7.5 Using advanced search functions to find specific data assets

IBM watsonx.data provides a suite of *advanced search functions* that significantly enhance the discoverability of data assets within a large catalog. Unlike basic search engines that rely on simple keyword matching, the IBM watsonx.data search engine incorporates advanced Boolean operators, filters, and facets to help users create highly specific queries by combining multiple search parameters to narrow down results with precision. With Boolean operators, users can include or exclude data sets based on criteria such as tags, metadata fields, or data set attributes.

Moreover, IBM watsonx.data supports *fuzzy search*, which accommodates minor discrepancies in search terms, such as spelling errors or slight variations in metadata entries. This approach helps ensure that even imperfect search terms yield relevant results, which reduce the likelihood of missing critical data sets due to human error.

For organizations with complex and highly specific search needs, IBM watsonx.data offers a rich set of APIs that can be integrated with external systems. With these APIs, you can build *custom search interfaces*, or you can automate routine search queries, which facilitate seamless workflows and enable data teams to access the data that they need without manual intervention.

7.6 Case study: Improving data discoverability for faster decision-making in the retail sector

Retail organizations operate in a highly dynamic and competitive environment, where timely and data-driven decisions are critical for success. These businesses deal with diverse data assets, which range from transactional sales data and customer feedback to supply chain metrics and promotional campaign analytics. The challenge is in managing vast and varied datasets and helping ensure that they are accessible, interpretable, and actionable for decision-makers across the organization.

IBM watsonx.data offers a solution by providing a centralized data catalog that organizes, classifies, and tags these data sets, making them easily discoverable. By automating much of the data preparation process, it helps retail organizations streamline workflows, enhance collaboration, and enable faster and more informed decision-making.

7.6.1 Use case: Unified data access across retail functions

Retailers frequently require insights that span multiple functional areas. For example, an organization might need to combine sales figures with supply chain data to pinpoint inventory restocking issues. At the same time, marketing teams might want to gauge the effectiveness of recent promotions on customer purchases.

With IBM watsonx.data, data sets from various domains are integrated into a single, comprehensive catalog, which helps ensure access and understanding of data regardless of its source. Metadata enrichment, automated tagging, and lineage tracking all contribute to this function.

IBM watsonx.data tackles a key challenge in retail data management: the manual classification and organization of data sets. It automates this process by using machine learning and NLP to identify patterns, classify content, and apply relevant business tags.

For example, transactional data sets can be automatically tagged with attributes like “Region: North America”, “Quarter: Q3”, and “Product Category: Electronics”. This automation reduces the burden on data stewards while helping ensure that data sets remain consistently organized. As a result, analysts can quickly locate data sets that meet their criteria, even in expansive catalogs.

IBM watsonx.data advanced search capabilities enable retailers to perform scenario-based analyses. Imagine a scenario where a retail executive wants to evaluate the effectiveness of a summer promotion. By using IBM watsonx.data, the executive can search for data sets that are related to summer sales, customer feedback during the promotional period, and inventory levels before and after the campaign.

By combining these data sets, the executive can identify trends, such as which products performed well during the promotion, whether customer satisfaction improved, and how effectively inventory was managed. These insights inform strategic decisions for future campaigns, such as product selection, pricing strategies, and marketing approaches.

Retail data is often diverse, encompassing structured formats (for example, relational databases with sales records), semi-structured formats (for example, JSON logs from e-commerce platforms), and unstructured formats (for example, customer reviews or social media posts). IBM watsonx.data is designed to handle this heterogeneity. For example, it can catalog structured sales data alongside unstructured customer feed-back. By indexing these data sets by using enriched metadata, IBM watsonx.data helps ensure that users can find and use both types of data in their analyses. Furthermore, it supports integrations with external tools, such as sentiment analysis platforms, which enable richer interpretations of unstructured data.

Retail organizations also face stringent compliance requirements, particularly regarding the handling of sensitive customer information. IBM watsonx.data provides robust security features, which include RBAC and data masking to ensure compliance with regulations such as GDPR and CCPA.

For example, personally identifiable information (PII), such as customer names and addresses, can be masked in search results while still allowing users to analyze non-sensitive attributes like purchase history or customer demographics. This capability helps ensure that analysts can work with data sets without risking privacy breaches or regulatory violations.

Although retail environments present unique challenges due to the diversity and scale of their data, IBM watsonx.data provides the necessary tools to address these complexities. By automating cataloging, enabling cross-functional data integration, and supporting advanced search functions, it transforms raw data into a strategic asset, which helps ensure that retail organizations are well equipped to make timely, data-driven decisions in a fast-paced and competitive market.

8

Marketing campaign analysis use case

This chapter dives into an example of a marketing campaign analysis at a financial services organization. In this use case, we use IBM watsonx.data and IBM watsonx.ai to ingest and store data, connect to and access data, visualize and explore data, and build machine learning (ML) models.

This chapter guides you through the following tasks: data ingestion, data connection and access, data exploration and visualization, and machine learning model development.

This chapter has the following sections:

8.1 Use case introduction

This chapter explores how a leading financial services company uses IBM watsonx.data and IBM watsonx.ai to unlock the power of marketing data.

This use case follows a marketing team at a financial services company. The team is looking to better understand the performance of their direct marketing campaigns for the bank's bank term deposit product (certificate of deposit).

A senior vice-president at the financial services company pulled together various individuals to help with this initiative: a data engineer, a data analyst, a data scientist, and various individuals from the business.

For this analysis, the team wants to use their environment. The organization is running IBM watsonx.ai and IBM watsonx.data on-premises on Red Hat OpenShift.

By analyzing the bank term deposit direct marketing campaign, the team aims to achieve the following goals:

- Assess the performance of direct marketing campaigns for bank term deposits and find ways to optimize their current marketing campaigns.
- Deepen their understanding of customer behavior and better understand the demographic of people they are reaching out to.
- Develop ways to predict whether a client will subscribe to a bank term deposit, and understand whether there are key factors that influence this decision.

In this use case, we use a basic data set to make the examples simple to follow and understand. This simplified bank marketing data set was retrieved from the UC Irvine Machine Learning Repository. To see the full data set, see Bank Marketing.

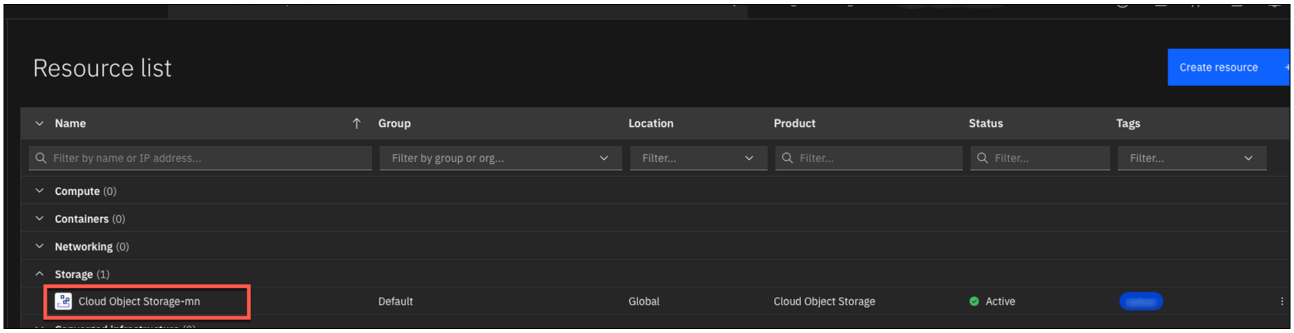
8.2 Data ingestion

Before data analysts and data scientists can analyze the marketing campaign data, the data engineer must make it available and accessible for analysis. The direct marketing campaign data for the bank term deposit is in an IBM Cloud Object Storage bucket in IBM Cloud. In this section, the data engineer identifies the location of the marketing campaign data and configures the necessary access with needed permissions, configures the storage and catalog in IBM watsonx.data, connects to the query engines in IBM watsonx.data, creates a schema in the catalog, and ingests data from the Cloud Object Storage bucket.

8.2.1 Locating the marketing campaign data

The data engineer talks with the application team that generated the marketing campaign data and finds that the data that is needed for analysis is stored in an IBM Cloud Object Storage bucket on IBM Cloud. To find that data, the data engineer completes the following steps:

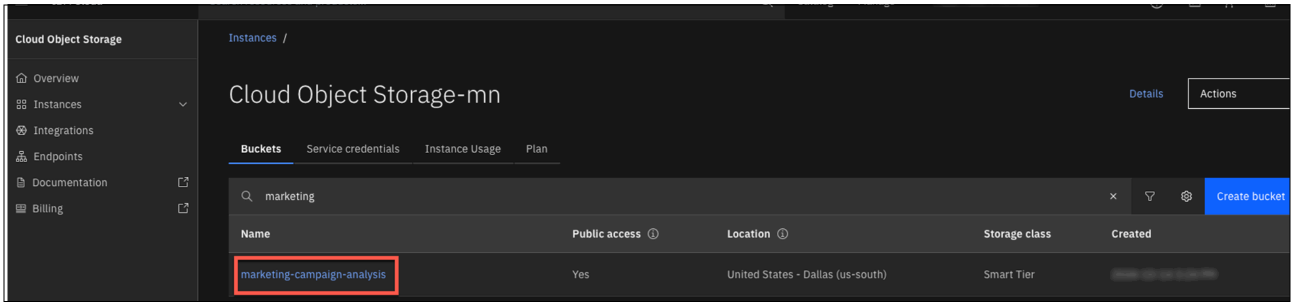
1. The data engineer navigates to the IBM Cloud Object Storage bucket by selecting **IBM Cloud** → **Resource List** → **Storage** → **Cloud Object Storage**, as shown in Figure 8-1 on page 103.



Name	Group	Location	Product	Status	Tags
Compute (0)					
Containers (0)					
Networking (0)					
Storage (1)					
Cloud Object Storage-mn	Default	Global	Cloud Object Storage	Active	[redacted]

Figure 8-1   Cloud Object Storage

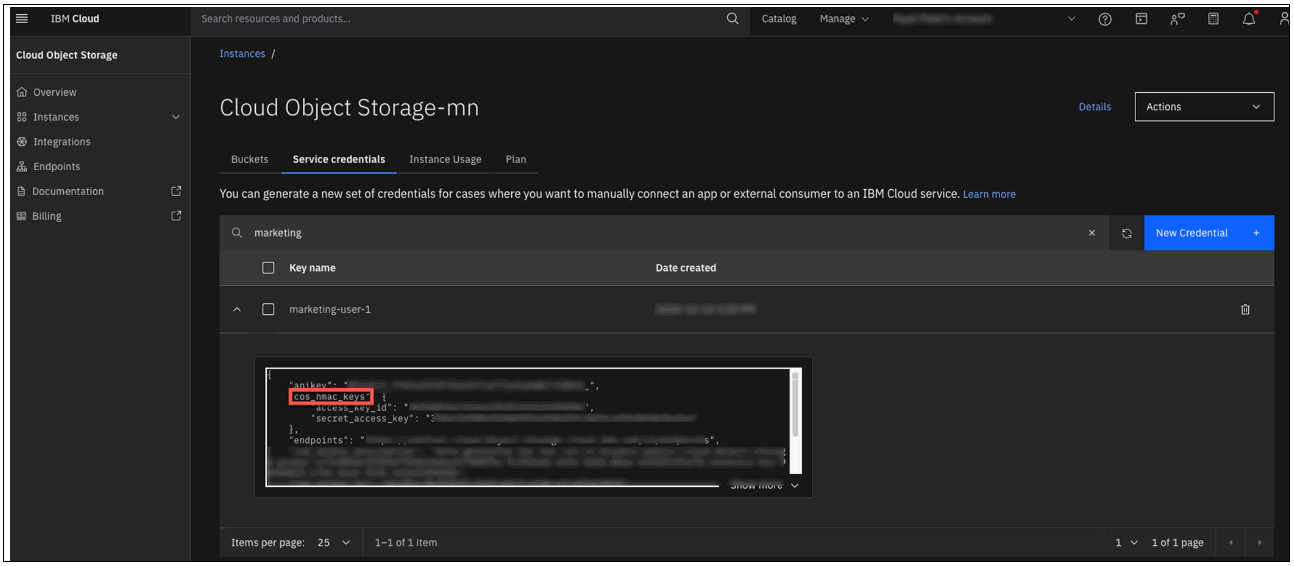
2. Then, the data engineer searches for the bucket that contains the marketing campaign performance data and selects the marketing-campaign-analysis bucket, as shown in Figure 8-2.



Name	Public access	Location	Storage class	Created
marketing-campaign-analysis	Yes	United States - Dallas (us-south)	Smart Tier	[redacted]

Figure 8-2   Searching for the bucket

3. The data engineer checks whether the service credentials for marketing-user-1 are present by selecting **Service credentials** → **marketing-user-1**. If not, the data engineer must assign the Manager role to the service credential and that the HMAC credentials are included so that the necessary information for the credentials is available to connect to in IBM watsonx.data. The data engineer notes the access key and secret because they know that this information will be needed later when configuring the IBM Cloud Object Storage storage in IBM watsonx.data. Figure 8-3 shows the service credentials.



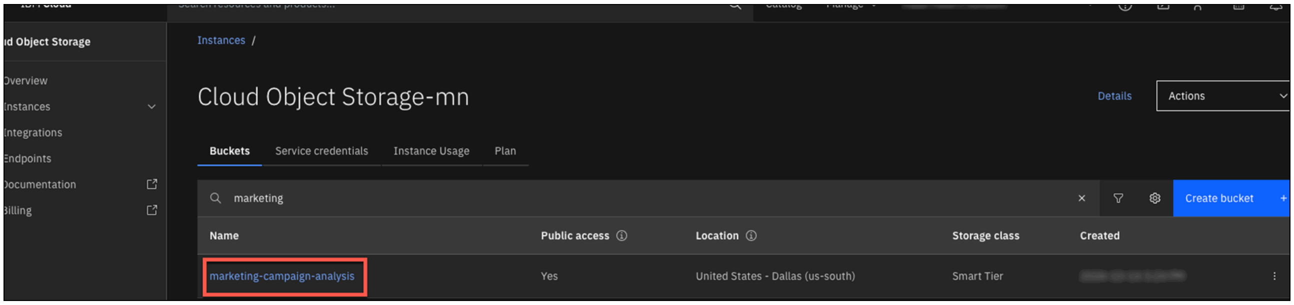
You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud service. Learn more

Key name	Date created
marketing-user-1	[redacted]

```
{
    "apikey": "[redacted]",
    "cos_hmac_keys": {
        "access_key_id": "[redacted]",
        "secret_access_key": "[redacted]
    },
    "endpoints": "[redacted]s",
[redacted]
```

Figure 8-3   Service credentials

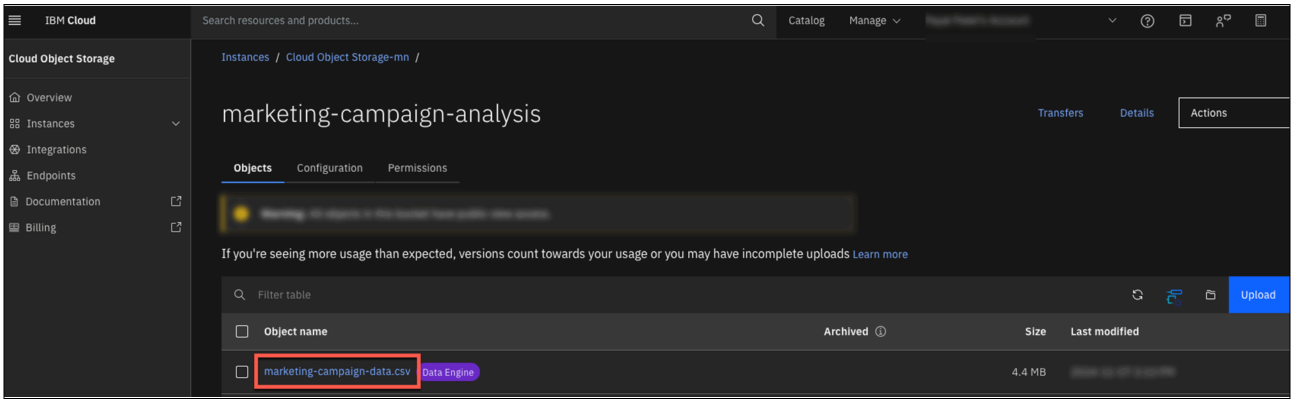
Once the service credentials are confirmed, the data engineer selects the **Buckets** tab and clicks the marketing-campaign-analysis IBM Cloud Object Storage bucket, as shown n Figure 8-4.



Name	Public access	Location	Storage class	Created
marketing-campaign-analysis	Yes	United States - Dallas (us-south)	Smart Tier	[redacted]

Figure 8-4 The marketing-campaign-analysis IBM Cloud Object Storage bucket

4. Within that bucket, the data engineer selects the **Objects** tab and confirms that the file that contains the marketing campaign data is present, as shown in Figure 8-5.

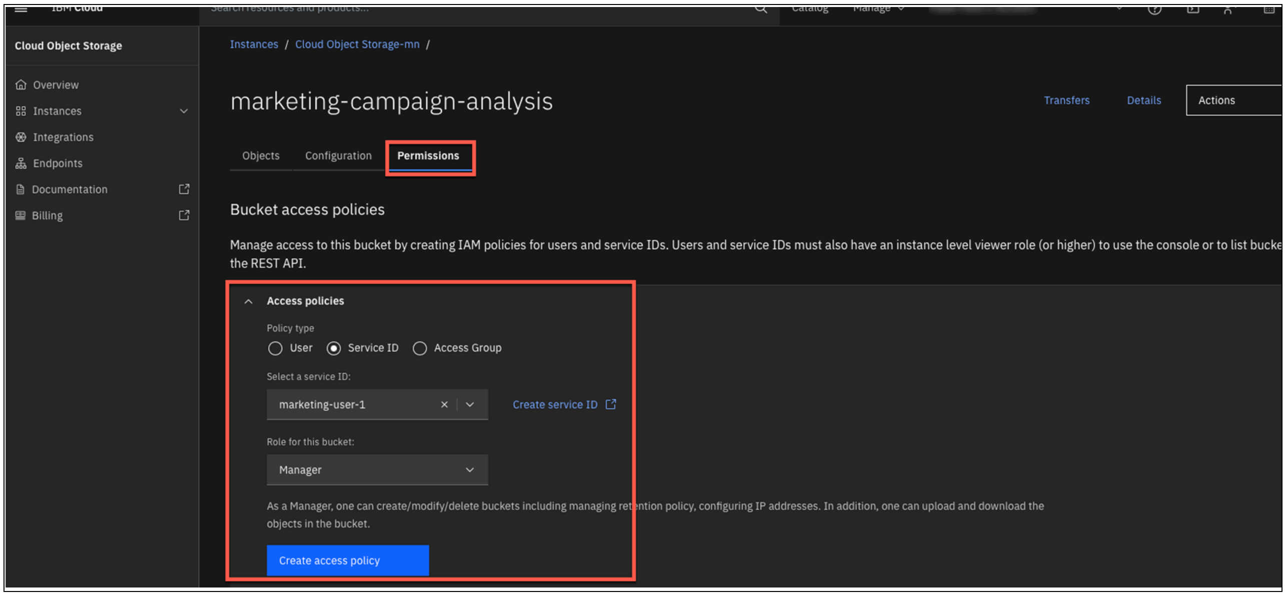


If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more

Object name	Archived	Size	Last modified
marketing-campaign-data.csv Data Engine		4.4 MB	[redacted]

Figure 8-5 The marketing-campaign-data.csv file

5. The data engineer adds permissions to the IBM Cloud Object Storage bucket for the service credential marketing-user-1. On the **Permissions** tab, they select **Access policies** → **Service ID** → **Create access policy** to create an access policy for the marketing-user-1 Service ID, with the Manager role assigned. This action is important because insufficient access can result in issues later when creating tables and schemas in IBM watsonx.data. Figure 8-6 on page 105 shows the Bucket access policies window.



marketing-campaign-analysis

Bucket access policies

Manage access to this bucket by creating IAM policies for users and service IDs. Users and service IDs must also have an instance level viewer role (or higher) to use the console or to list bucke the REST API.

Access policies

Policy type

○ User ● Service ID ○ Access Group

Select a service ID:

marketing-user-1

Create service ID

Role for this bucket:

Manager

As a Manager, one can create/modify/delete buckets including managing retention policy, configuring IP addresses. In addition, one can upload and download the objects in the bucket.

Create access policy

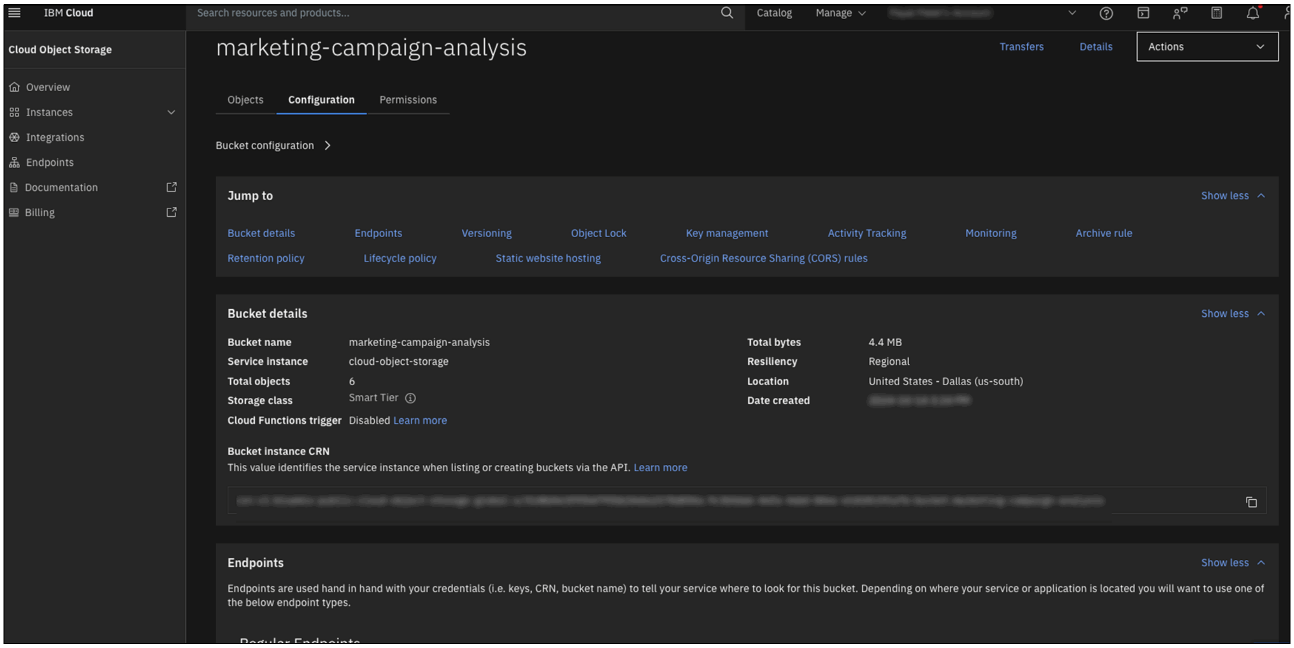
Figure 8-6 Creating an access policy

6. On the **Configuration** tab, the data engineer collects the following information that will be needed later to establish a connection to the IBM Cloud Object Storage in IBM watsonx.data:

- Bucket name: marketing-campaign-analysis
- Location (that is, Region): United States - Dallas (us-south)
- Public endpoint: s3.us-south.cloud-object-storage.appdomain.cloud

> **Note:** An access key and secret are also required when establishing the connection in IBM watsonx.data. The data engineer noted this information in step 3 on page 103.

Figure 8-7 shows the endpoints of the bucket.



marketing-campaign-analysis

Jump to

Bucket details Endpoints Versioning Object Lock Key management Activity Tracking Monitoring Archive rule
Retention policy Lifecycle policy Static website hosting Cross-Origin Resource Sharing (CORS) rules

Bucket details

Bucket name	marketing-campaign-analysis	Total bytes	4.4 MB
Service instance	cloud-object-storage	Resiliency	Regional
Total objects	6	Location	United States - Dallas (us-south)
Storage class	Smart Tier ⓘ	Date created	[redacted]
Cloud Functions trigger	Disabled Learn more		

Bucket instance CRN

This value identifies the service instance when listing or creating buckets via the API. Learn more

[redacted]

Endpoints

Endpoints are used hand in hand with your credentials (i.e. keys, CRN, bucket name) to tell your service where to look for this bucket. Depending on where your service or application is located you will want to use one of the below endpoint types.

Regular Endpoints

Figure 8-7 Bucket details: Endpoints

8.2.2 Setting up the internal spark engine

Before ingesting data into IBM watsonx.data from the IBM Cloud Object Storage bucket, the data engineer configures the native Spark engine on IBM watsonx.data. To do so, the data engineer completes the following steps:

1. The data engineer logs in to IBM watsonx.data and navigates to the Infrastructure manager window and selects **Add component** to add the Spark engine, as shown in Figure 8-8.

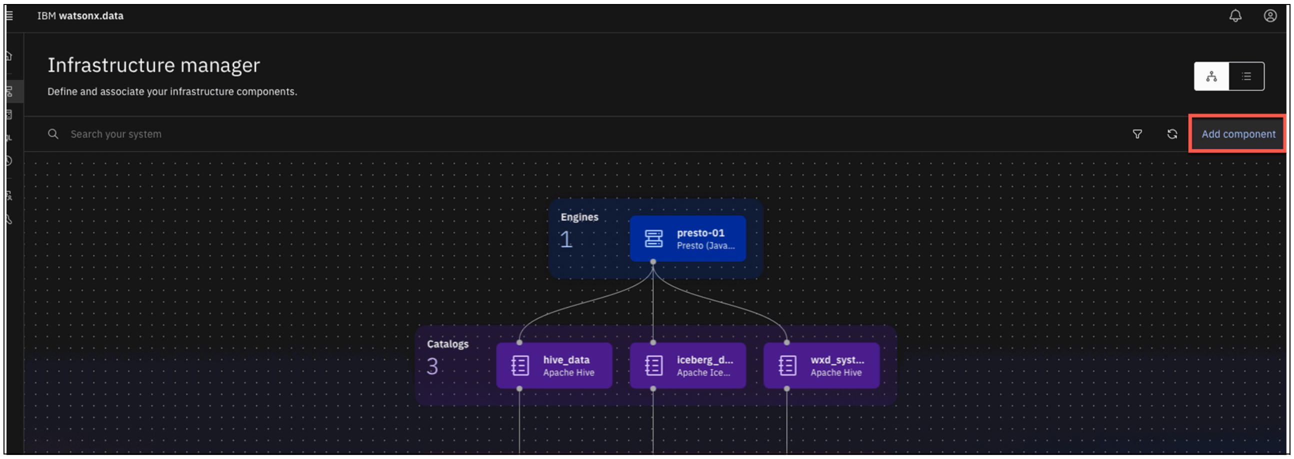


Figure 8-8 Add component

2. Under Engines, the data engineer clicks **IBM Spark**, and then clicks **Next**, as shown in Figure 8-9.

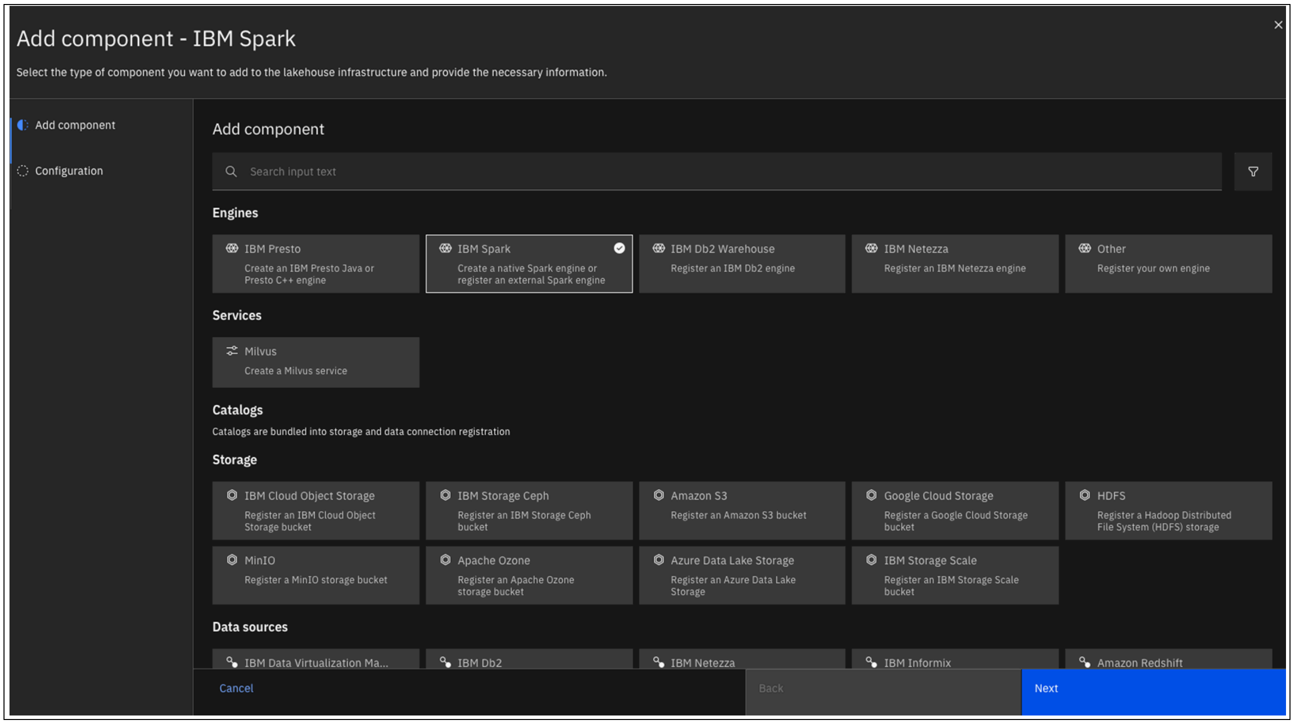
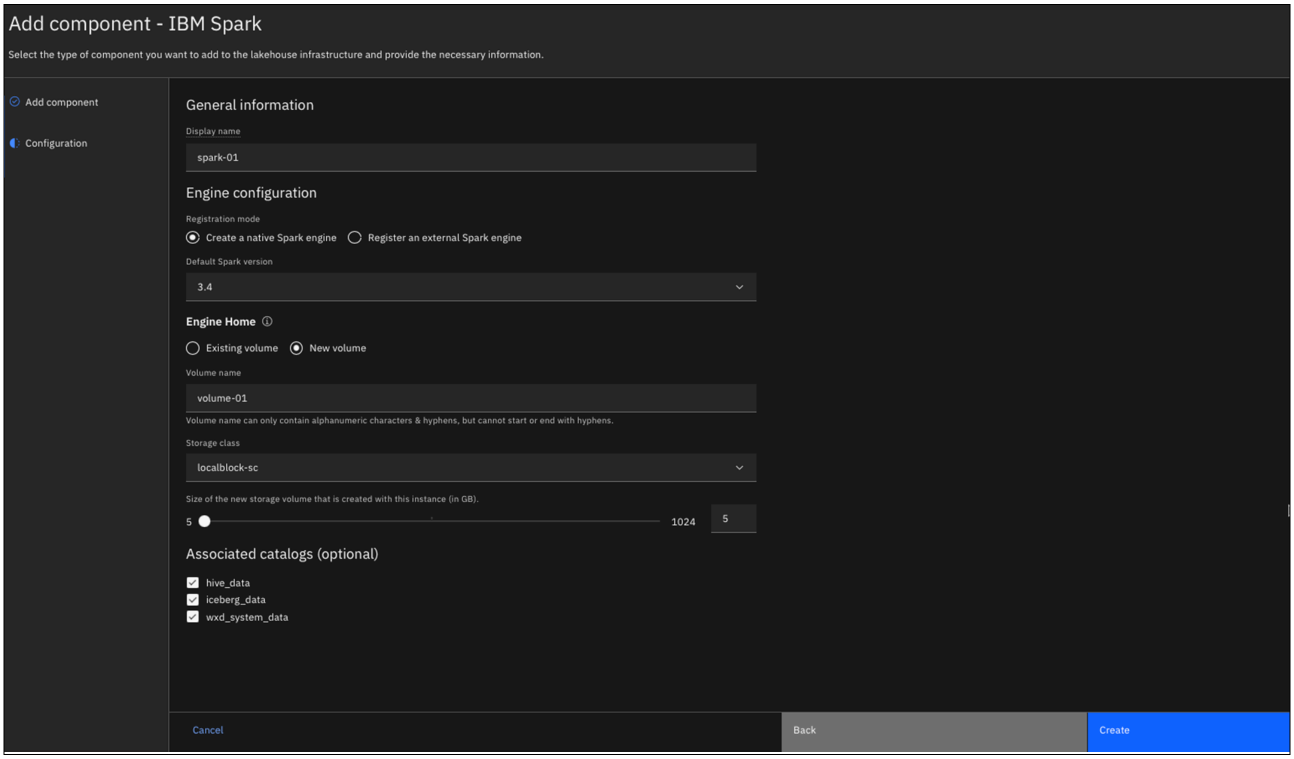

Add component - IBM Spark
Select the type of component you want to add to the lakehouse infrastructure and provide the necessary information.
Add component
Configuration
Add component
Search input text
Engines
IBM Presto
Create an IBM Presto Java or Presto C++ engine
IBM Spark
Create a native Spark engine or register an external Spark engine
IBM Db2 Warehouse
Register an IBM Db2 engine
IBM Netezza
Register an IBM Netezza engine
Other
Register your own engine
Services
Milvus
Create a Milvus service
Catalogs
Catalogs are bundled into storage and data connection registration
Storage
IBM Cloud Object Storage
Register an IBM Cloud Object Storage bucket
IBM Storage Ceph
Register an IBM Storage Ceph bucket
Amazon S3
Register an Amazon S3 bucket
Google Cloud Storage
Register a Google Cloud Storage bucket
HDFS
Register a Hadoop Distributed File System (HDFS) storage
MinIO
Register a MinIO storage bucket
Apache Ozone
Register an Apache Ozone storage bucket
Azure Data Lake Storage
Register an Azure Data Lake Storage
IBM Storage Scale
Register an IBM Storage Scale bucket
Data sources
IBM Data Virtualization Ma...
IBM Db2
IBM Netezza
IBM Informix
Amazon Redshift
Cancel
Back
Next

Figure 8-9 Add component: IBM Spark

3. Before creating the IBM Spark engine instance, the data engineer adds the engine details, which includes information to create a volume for the engine and catalogs to associate to the engine by selecting **General Information** → **Engine Configuration** → **Associate catalogs** → **Create**, as shown in Figure 8-10.



Add component - IBM Spark

Select the type of component you want to add to the lakehouse infrastructure and provide the necessary information.

General information

Display name: spark-01

Engine configuration

Registration mode: ● Create a native Spark engine ○ Register an external Spark engine

Default Spark version: 3.4

Engine Home

○ Existing volume ● New volume

Volume name: volume-01

Volume name can only contain alphanumeric characters & hyphens, but cannot start or end with hyphens.

Storage class: localblock-sc

Size of the new storage volume that is created with this instance (in GB).

5 1024 5

Associated catalogs (optional)

☑ hive_data
☑ iceberg_data
☑ wxd_system_data

Figure 8-10 Add component: IBM Spark

The new Spark engine takes a few minutes to provision. Once provisioned, the data engineer can view and access this engine from the Infrastructure manager page, as shown in Figure 8-11.

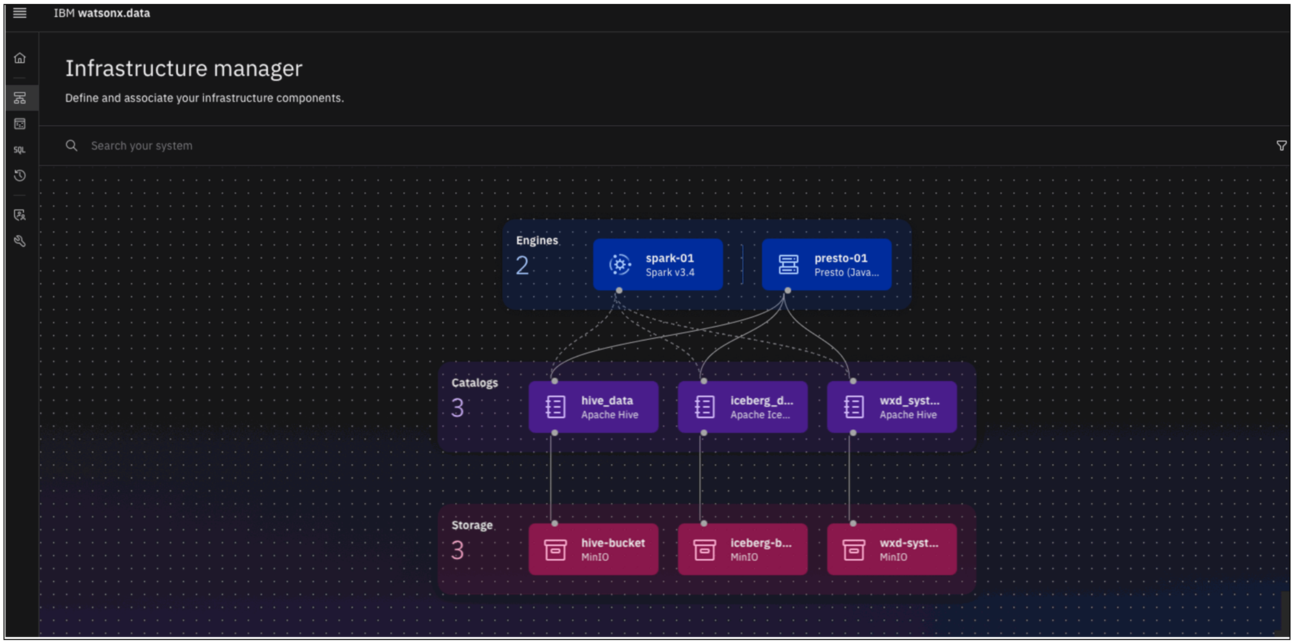


Figure 8-11 Infrastructure manager view with IBM Spark engine spark-01 added

8.2.3 Configure storage and catalog in IBM watsonx.data

After configuring the native Spark engine and reviewing the Infrastructure manager page, the data engineer finds that they need to configure the IBM Cloud Object Storage bucket and create a catalog in IBM watsonx.data for the marketing campaign data. To do so, they complete the following steps:

1. From the Infrastructure manager window, the data engineer selects **Add component** to add the IBM Cloud Object Storage instance, as shown in Figure 8-12.

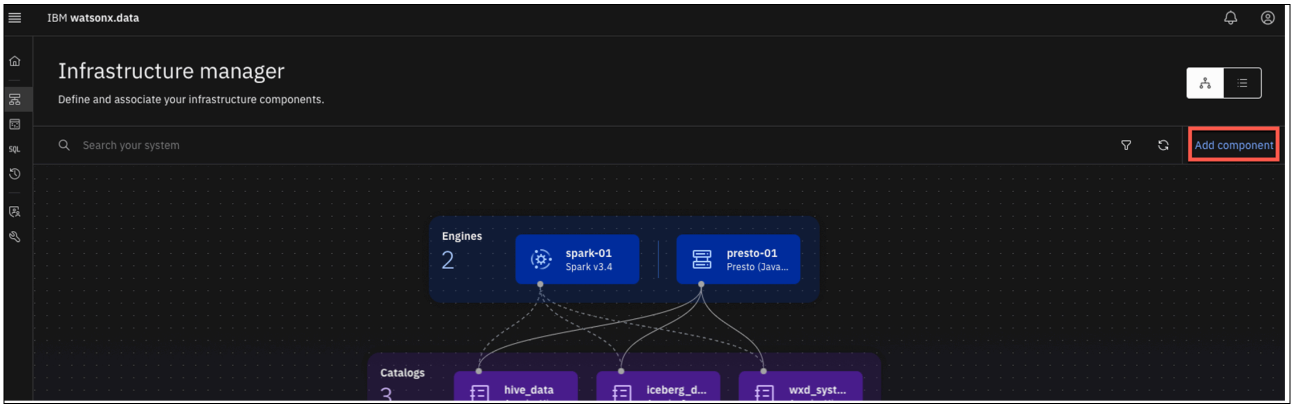
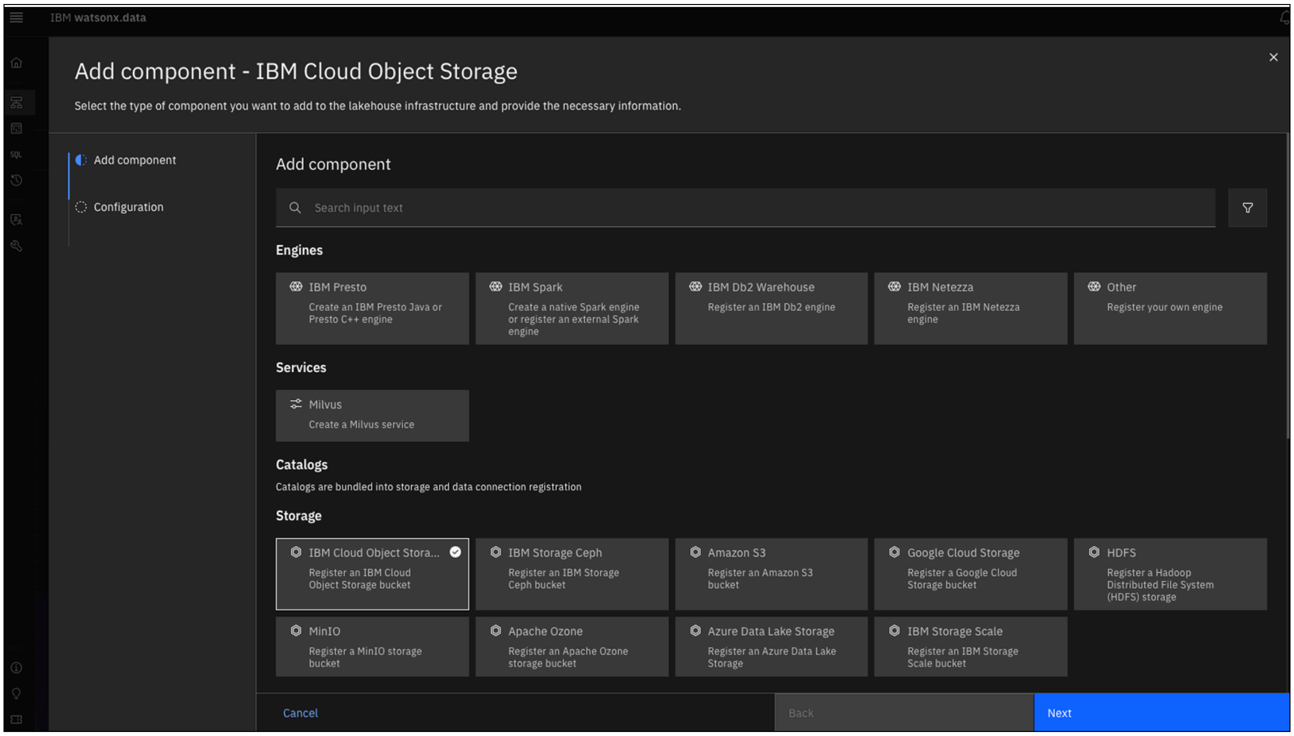


Figure 8-12 Infrastructure Manager: Add component

2. Under the Storage section, the data engineer clicks **IBM Cloud Object Storage**, and then clicks **Next** to continue to the configuration details, as shown in Figure 8-13 on page 109.


IBM watsonx.data

Add component - IBM Cloud Object Storage

Select the type of component you want to add to the lakehouse infrastructure and provide the necessary information.

Add component

Configuration

Add component

Search input text

Engines

IBM Presto — Create an IBM Presto Java or Presto C++ engine

IBM Spark — Create a native Spark engine or register an external Spark engine

IBM Db2 Warehouse — Register an IBM Db2 engine

IBM Netezza — Register an IBM Netezza engine

Other — Register your own engine

Services

Milvus — Create a Milvus service

Catalogs

Catalogs are bundled into storage and data connection registration

Storage

IBM Cloud Object Stora... ✓ — Register an IBM Cloud Object Storage bucket

IBM Storage Ceph — Register an IBM Storage Ceph bucket

Amazon S3 — Register an Amazon S3 bucket

Google Cloud Storage — Register a Google Cloud Storage bucket

HDFS — Register a Hadoop Distributed File System (HDFS) storage

MinIO — Register a MinIO storage bucket

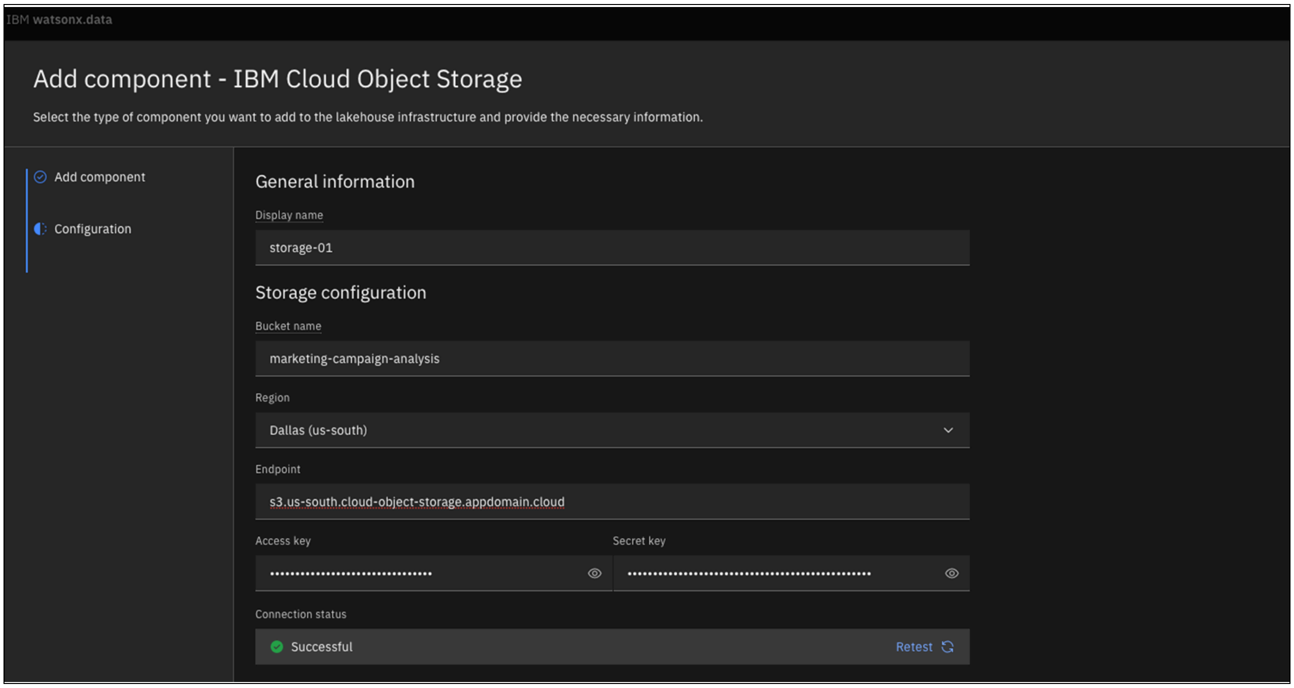
Apache Ozone — Register an Apache Ozone storage bucket

Azure Data Lake Storage — Register an Azure Data Lake Storage

IBM Storage Scale — Register an IBM Storage Scale bucket

Figure 8-13 IBM Cloud Object Storage

3. Using the details that were saved from the IBM Cloud Object Storage instance, as highlighted in 8.2, “Data ingestion” on page 102, the data engineer completes the Storage configuration details, which include Bucket name, Region, Endpoint, Access key, and Secret key, as shown in Figure 8-14.


IBM watsonx.data

Add component - IBM Cloud Object Storage

Select the type of component you want to add to the lakehouse infrastructure and provide the necessary information.

Add component

Configuration

General information

Display name: storage-01

Storage configuration

Bucket name: marketing-campaign-analysis

Region: Dallas (us-south)

Endpoint: s3.us-south.cloud-object-storage.appdomain.cloud

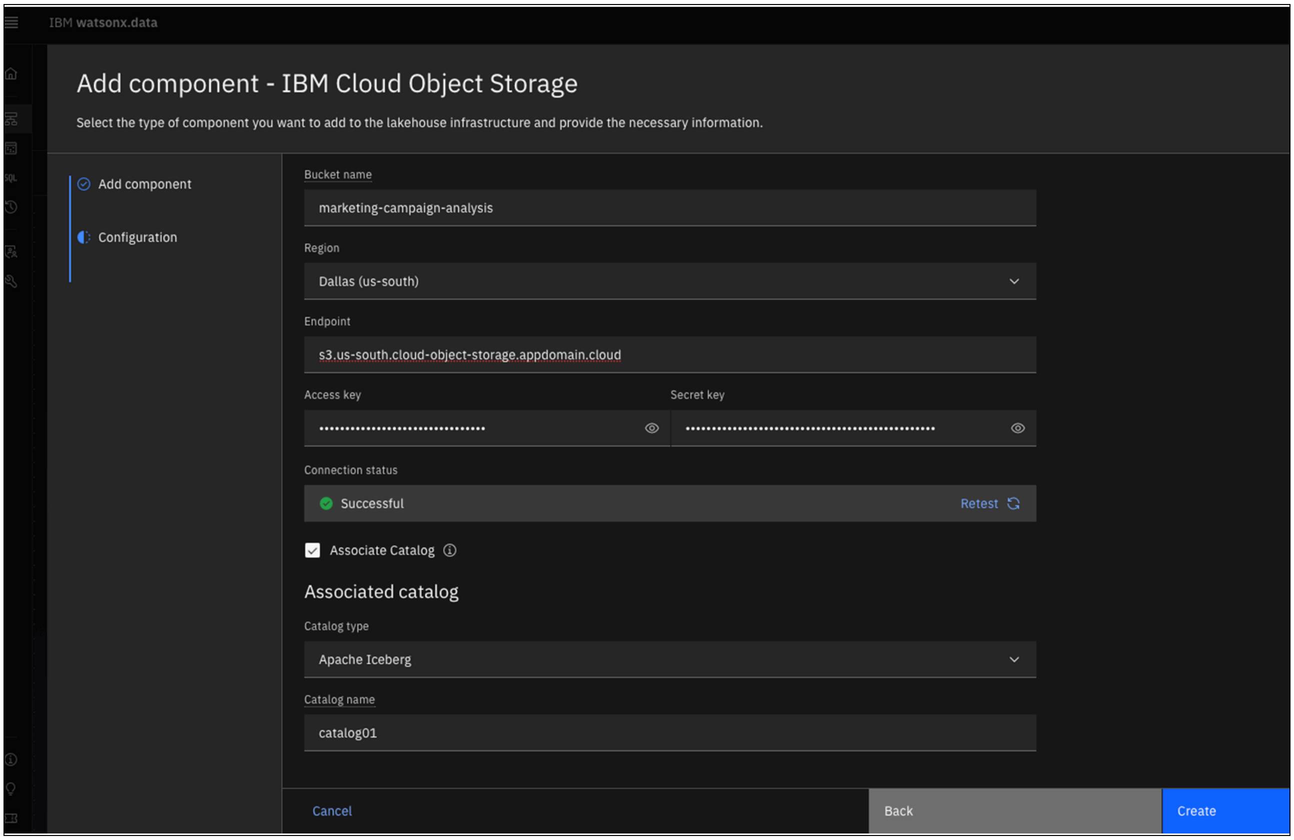
Access key: ••••••••••••••••••••••••••••••••••

Secret key: ••

Connection status: Successful

Figure 8-14 Storage configuration

4. Then, they create a catalog and associate it to the IBM Cloud Object Storage component, and then click **Create**, as shown in Figure 8-15.



IBM watsonx.data

Add component - IBM Cloud Object Storage

Select the type of component you want to add to the lakehouse infrastructure and provide the necessary information.

Bucket name: marketing-campaign-analysis

Region: Dallas (us-south)

Endpoint: s3.us-south.cloud-object-storage.appdomain.cloud

Access key: ••••••••••••••••••••••••••••••••••

Secret key: •••

Connection status: Successful

☑ Associate Catalog

Associated catalog

Catalog type: Apache Iceberg

Catalog name: catalog01

Figure 8-15 Associated catalog - Create

The new storage component and the new catalog take a few minutes to provision. Once provisioned, the data engineer can see them in the Infrastructure manager view, as shown in Figure 8-16 on page 111.

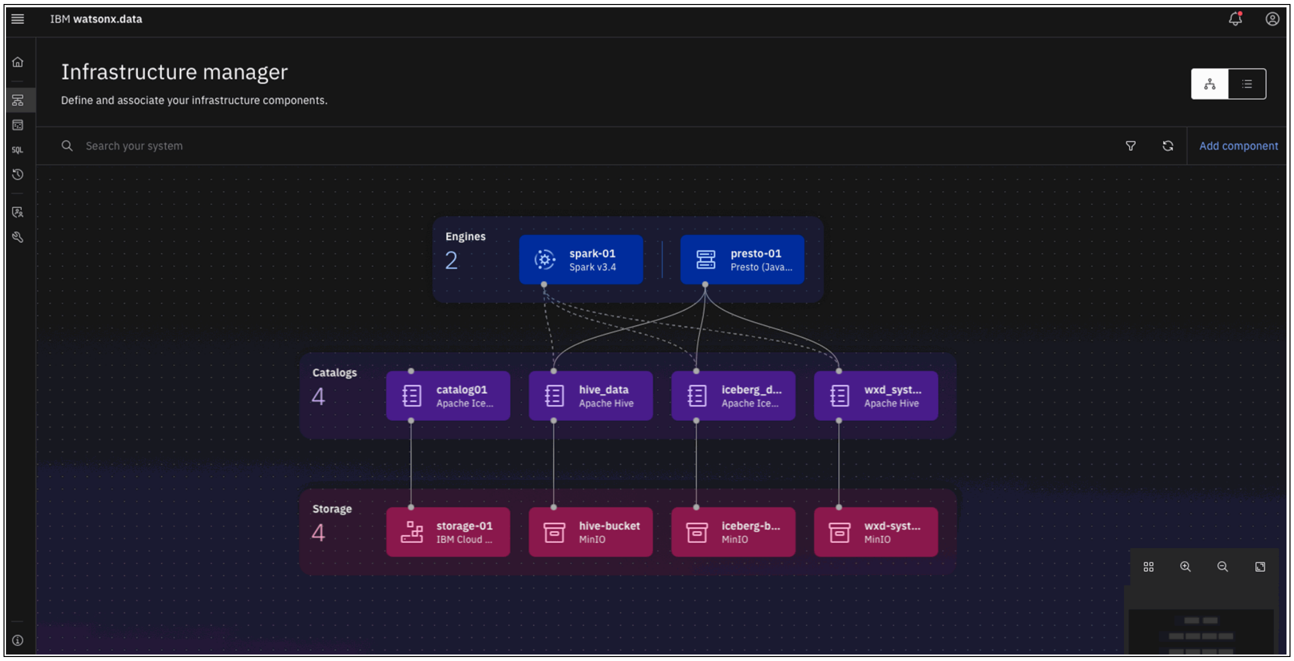


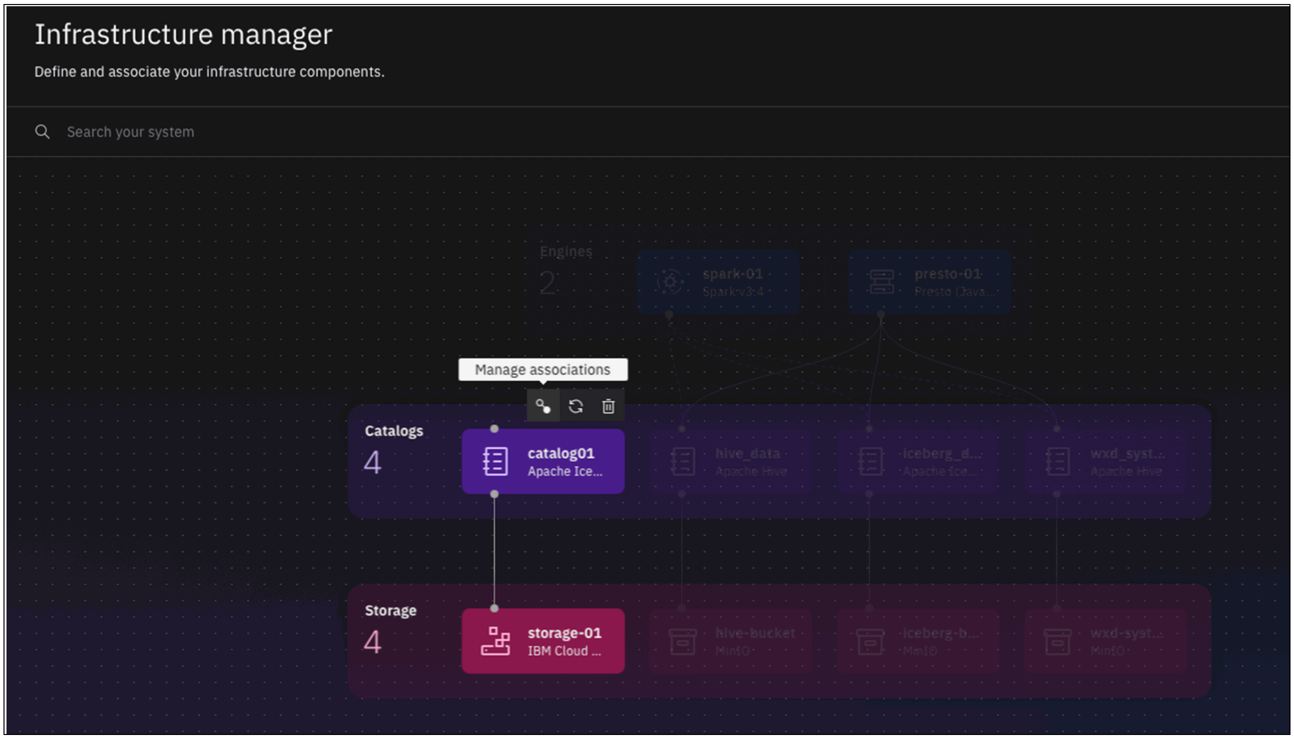
Figure 8-16 Infrastructure manager with new storage component and catalog created

8.2.4 Connecting query engines to catalog and storage in IBM watsonx.data

After the IBM Cloud Object Storage instance is configured in IBM watsonx.data and the new catalog is created, the data engineer connects these objects to the query engines in IBM watsonx.data by using the Infrastructure manager.

To do so, they complete the following steps:

1. The data engineer hovers their cursor over the newly created catalog catalog01 and clicks **Manage associations**, as shown in Figure 8-17.

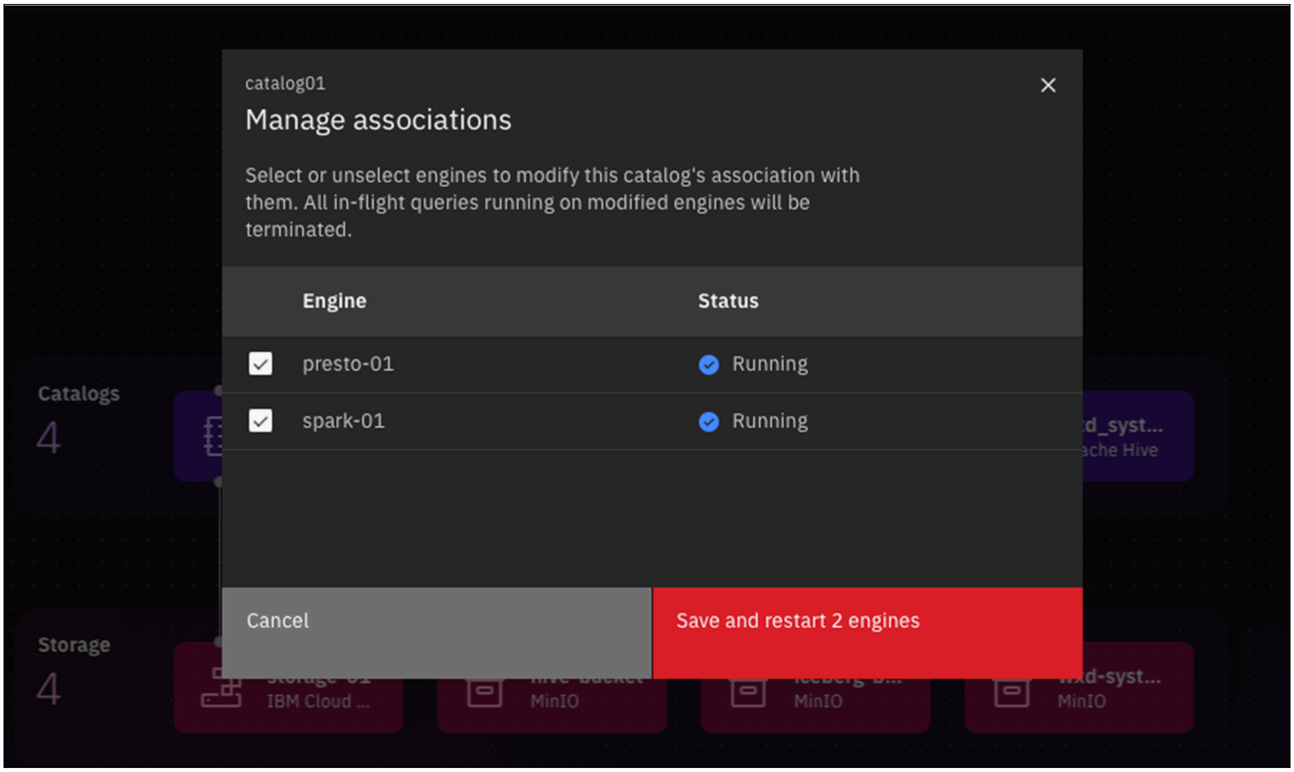


Infrastructure manager

Define and associate your infrastructure components.

Figure 8-17   Infrastructure manager: Manage associations

2. The data engineer selects the two engines that are available in their IBM watsonx.data instance, presto-01 and spark-01, and clicks **Save and restart 2 engines**, as shown in Figure 8-18 on page 113.



catalog01

Manage associations

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

	Engine	Status
☑	presto-01	Running
☑	spark-01	Running

Figure 8-18   Manage associations: Save and restart 2 engines

It takes a few minutes for the engines to restart with the new changes and associations. Once the process is complete, the data engineer can see the associations in the Infrastructure manager, as shown in Figure 8-19.

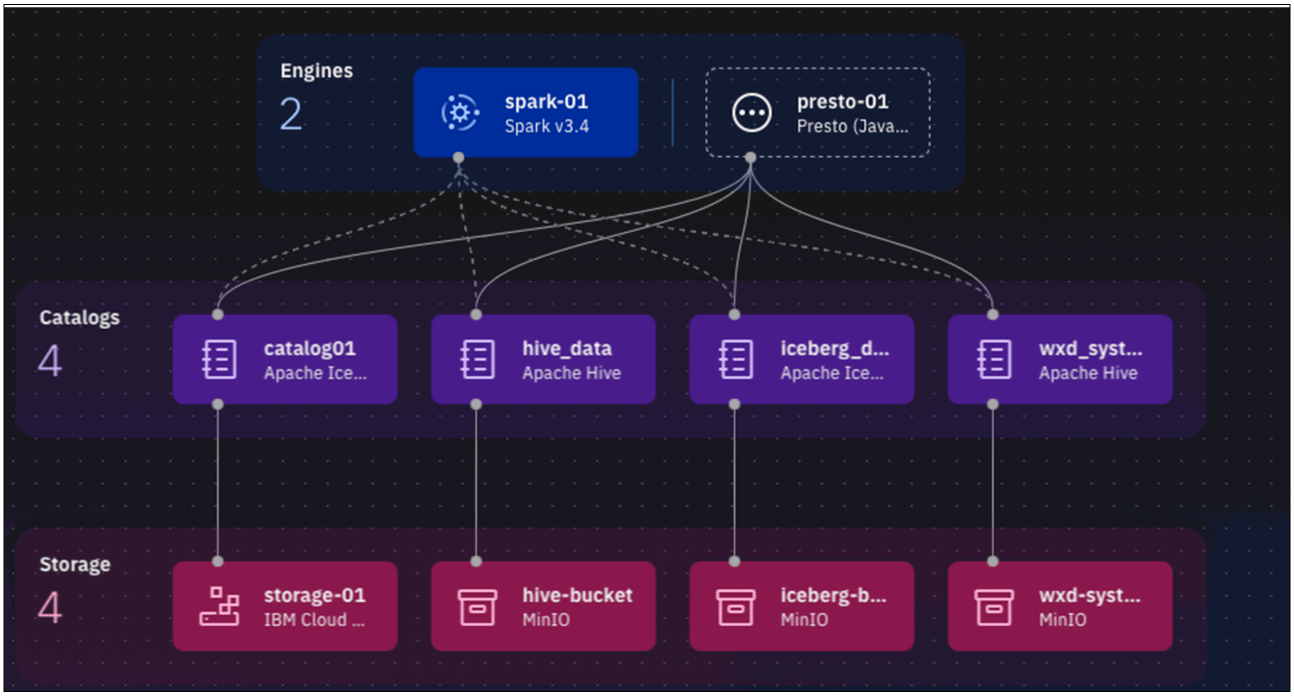


Figure 8-19   Infrastructure manager

8.2.5 Creating a schema in the catalog

The data engineer is ready to create a schema in the catalog that was created in 8.2.3, “Configure storage and catalog in IBM watsonx.data” on page 108. This new schema is used to ingest the marketing campaign data from the IBM Cloud Object Storage bucket. To do this task, they complete the following steps:

1. On the Data manager page, the data engineer selects **Create** → **Create schema**, as shown in Figure 8-20.

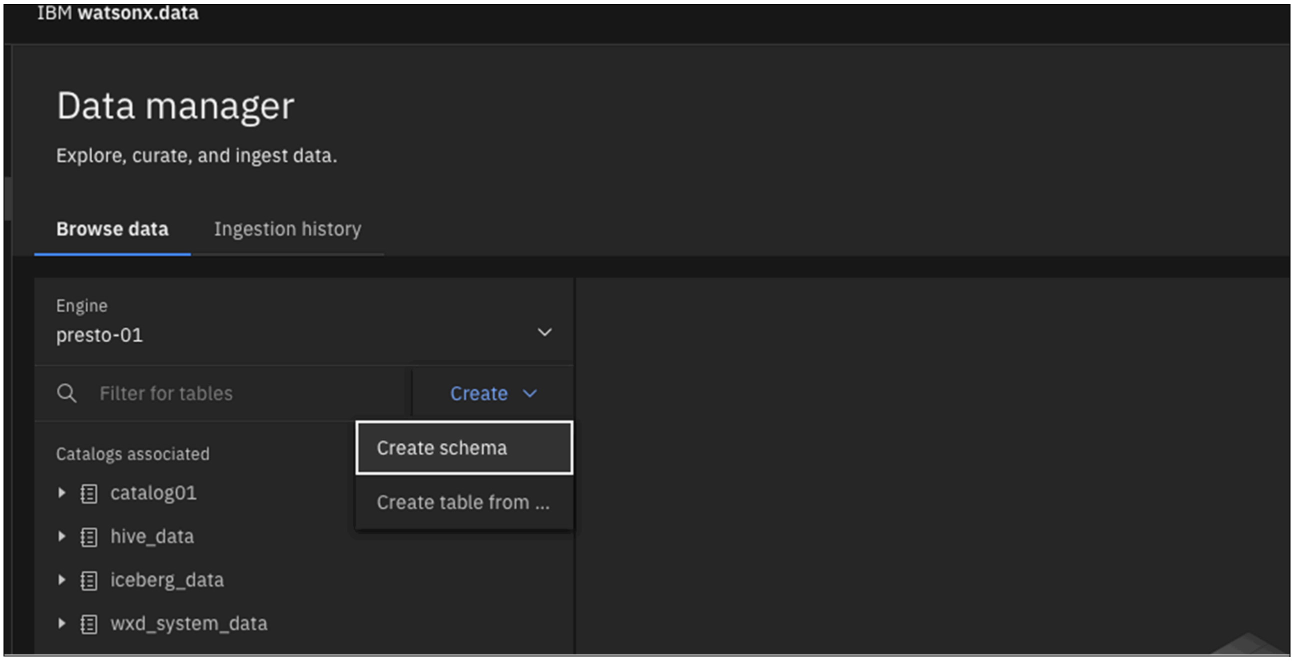
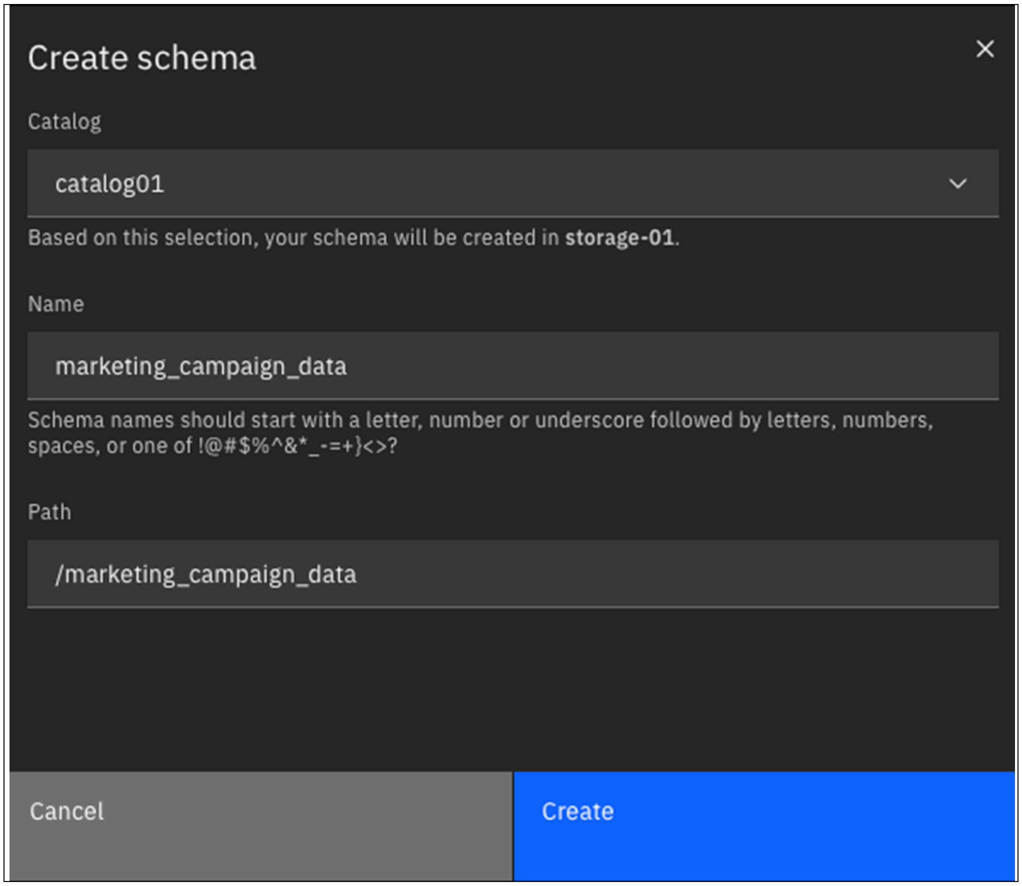


Figure 8-20 Create schema

2. The data engineer defines the Catalog, Name, and Path, as shown in Figure 8-21 on page 115.



Create schema

Catalog

catalog01

Based on this selection, your schema will be created in **storage-01**.

Name

marketing_campaign_data

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

Path

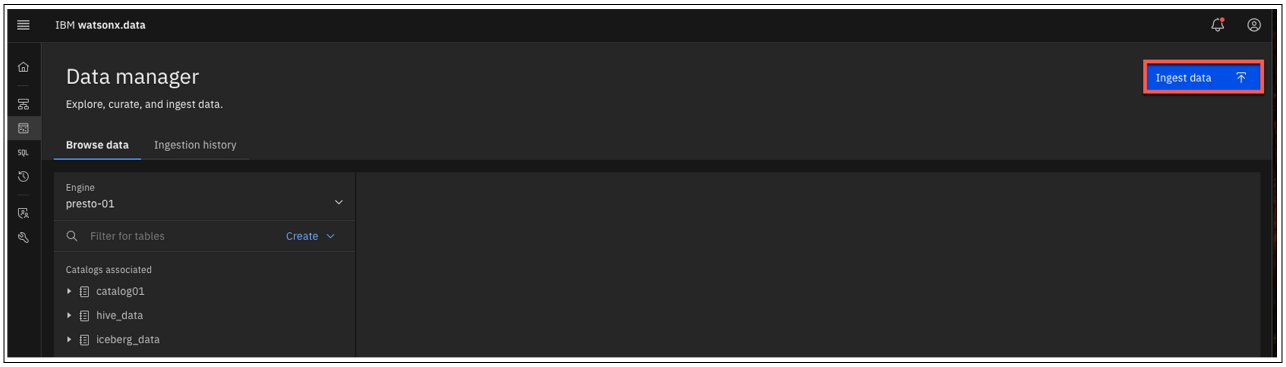
/marketing_campaign_data

Figure 8-21 Create schema

8.2.6 Ingesting data from an IBM Cloud Object Storage bucket

The data engineer proceeds to ingest the marketing campaign data from the IBM Cloud Object Storage bucket. To do so, they complete the following steps:

1. From the Data manager page, they click **Ingest data** to begin, as shown in Figure 8-22.



Data manager

Explore, curate, and ingest data.

Engine

presto-01

Catalogs associated

- catalog01
- hive_data
- iceberg_data

Figure 8-22 Ingest data

2. To ingest data, the data engineer has three options to select from: Local System, Storages, and Databases. Because the data engineer already configured the IBM Cloud Object Storage bucket as a storage component by using the Infrastructure manager in IBM watsonx.data, they click **Storages**, as shown in Figure 8-23.

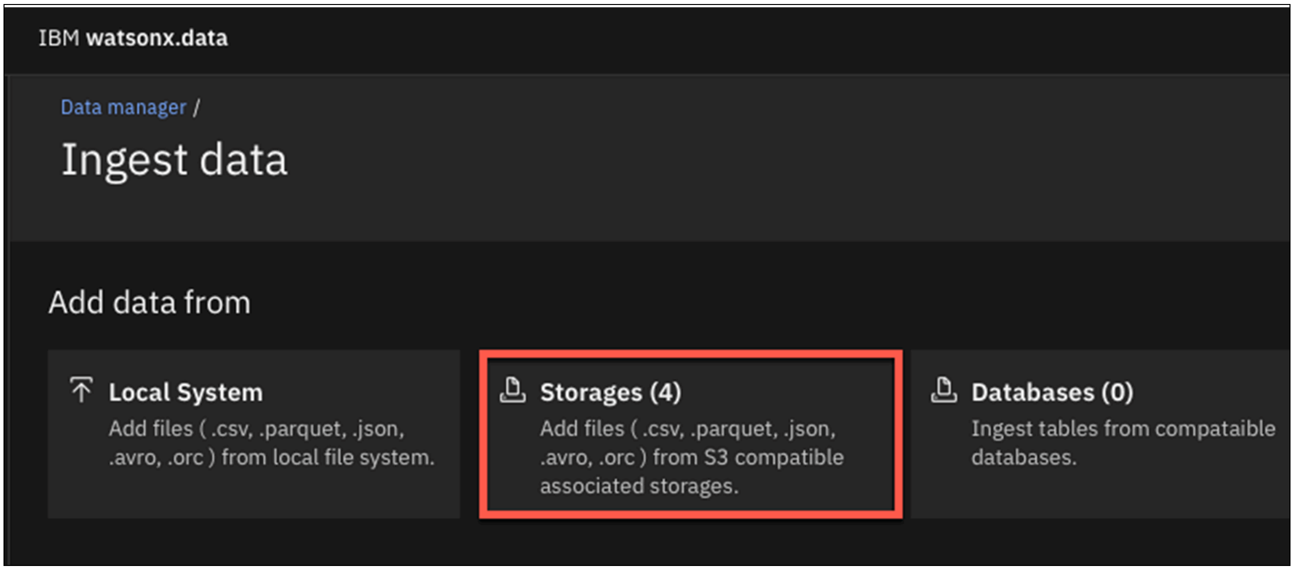
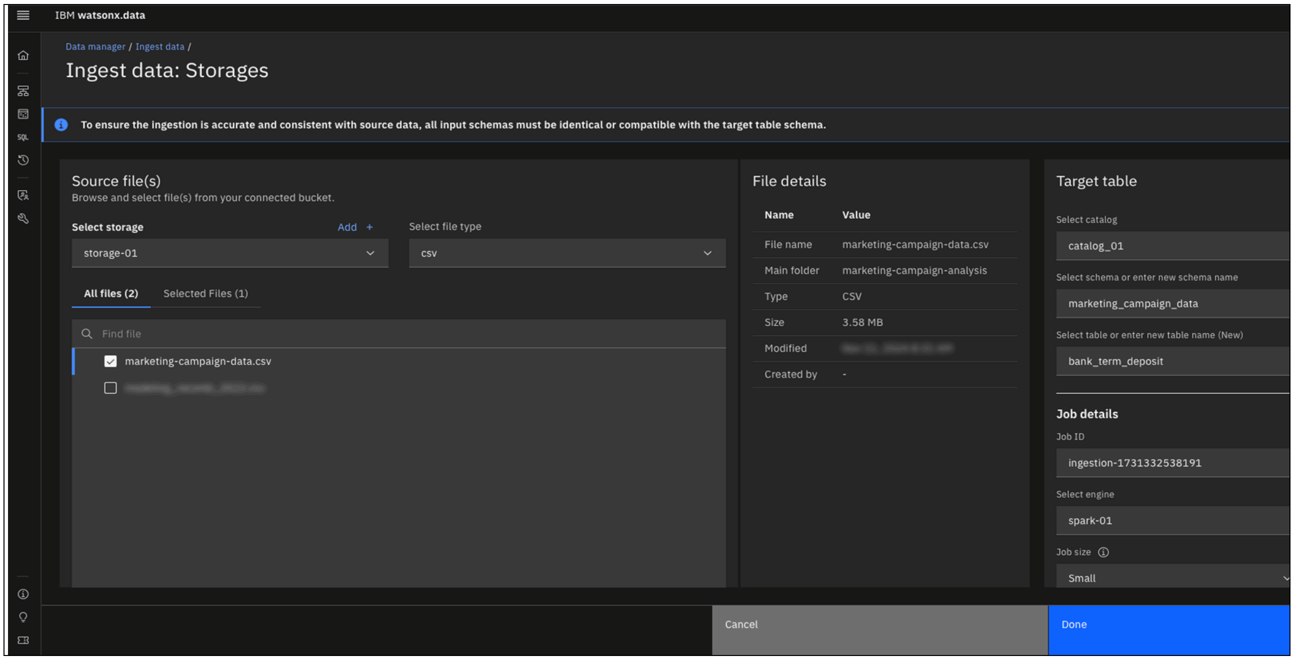

IBM watsonx.data
Data manager /
Ingest data
Add data from
Local System
Add files (.csv, .parquet, .json, .avro, .orc) from local file system.
Storages (4)
Add files (.csv, .parquet, .json, .avro, .orc) from S3 compatible associated storages.
Databases (0)
Ingest tables from compataible databases.

Figure 8-23 Ingest data - Storages

3. The data engineer selects the source file and defines the target table before clicking **Done**, as shown in Figure 8-24 on page 117. The source file is selected from the storage-01 component that was defined in 8.2.3, “Configure storage and catalog in IBM watsonx.data” on page 108. The target table points to the catalog catalog_01, which was created in 8.2.3, “Configure storage and catalog in IBM watsonx.data” on page 108, and the schema marketing_campaign_data, which was created in 8.2.5, “Creating a schema in the catalog” on page 114. The data engineer adds a new name for the table and then clicks **Done**.



IBM watsonx.data

Data manager / Ingest data /

Ingest data: Storages

To ensure the ingestion is accurate and consistent with source data, all input schemas must be identical or compatible with the target table schema.

Source file(s)

Browse and select file(s) from your connected bucket.

Select storage: storage-01

Select file type: csv

All files (2) | Selected Files (1)

☑ marketing-campaign-data.csv

□ [redacted]

File details

Name	Value
File name	marketing-campaign-data.csv
Main folder	marketing-campaign-analysis
Type	CSV
Size	3.58 MB
Modified	[redacted]
Created by	-

Target table

Select catalog: catalog_01

Select schema or enter new schema name: marketing_campaign_data

Select table or enter new table name (New): bank_term_deposit

Job details

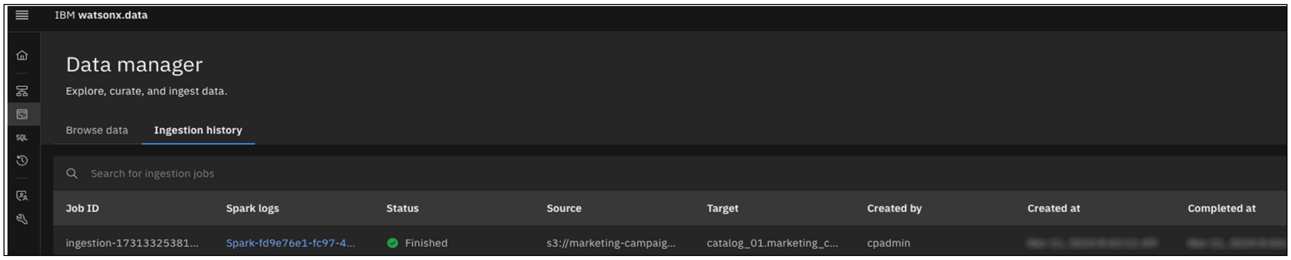
Job ID: ingestion-1731332538191

Select engine: spark-01

Job size: Small

Figure 8-24 Source files: Target table

The ingestion job typically requires a few minutes to finalize its execution. The data engineer can see the status of the job on the **Ingestion history** tab within the Data manager page, as shown in Figure 8-25.



IBM watsonx.data

Data manager

Explore, curate, and ingest data.

Browse data | Ingestion history

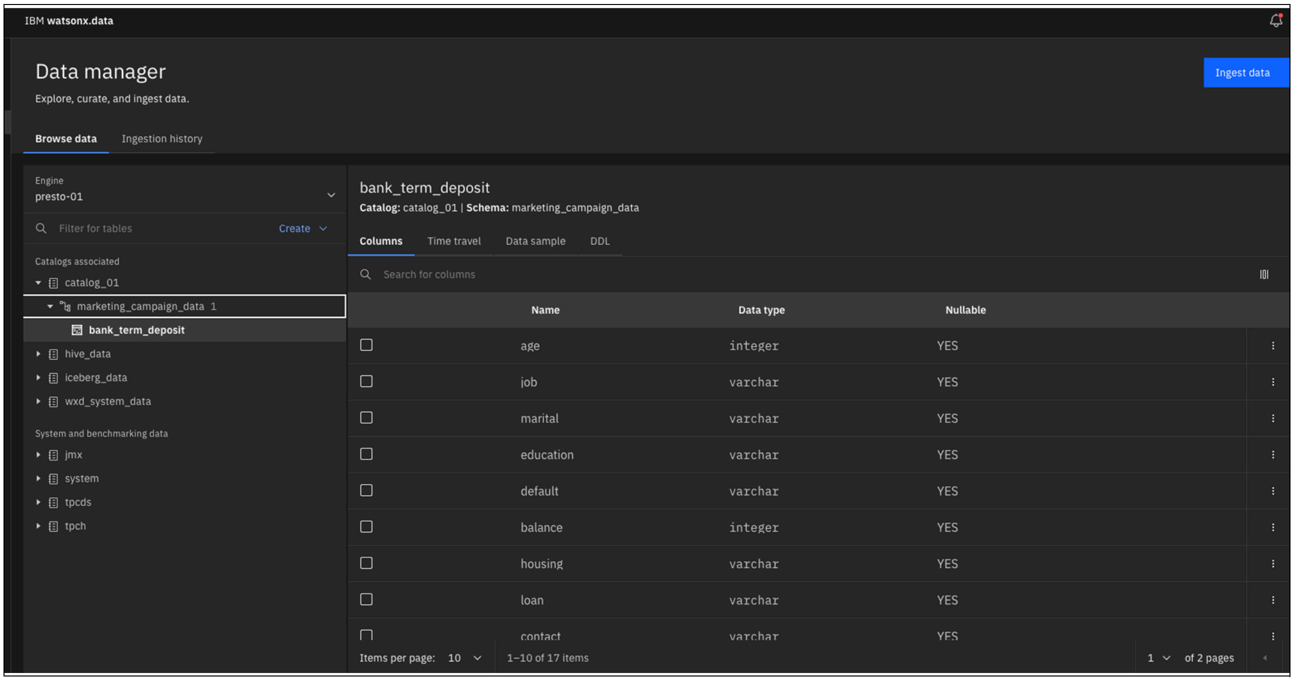
Job ID	Spark logs	Status	Source	Target	Created by	Created at	Completed at
ingestion-17313325381...	Spark-fd9e76e1-fc97-4...	Finished	s3://marketing-campaig...	catalog_01.marketing_c...	cpadmin	[redacted]	[redacted]

Figure 8-25 Ingestion history

8.2.7 Verifying the data in the schema

Once the data ingestion job completes, the data engineer verifies that the data is in the correct table and schema. To do so, they complete the following steps:

1. From the Data manager window, they select the catalog, schema, and table that are defined in the ingestion job (select **catalog_01** → **marketing_campaign_data** → **bank_term_deposit** → **Columns**), as shown in Figure 8-26. The **Columns** tab on the right shows the columns in the table, including the data type.



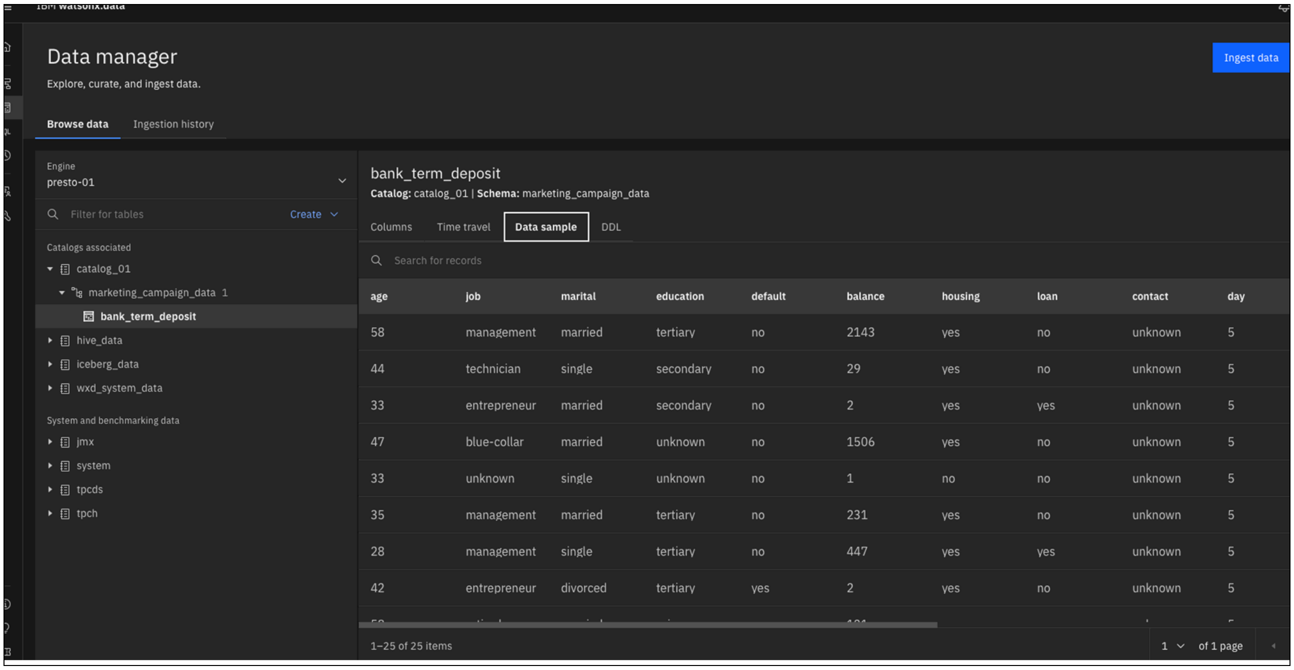
bank_term_deposit
Catalog: catalog_01 | **Schema:** marketing_campaign_data

	Name	Data type	Nullable
□	age	integer	YES
□	job	varchar	YES
□	marital	varchar	YES
□	education	varchar	YES
□	default	varchar	YES
□	balance	integer	YES
□	housing	varchar	YES
□	loan	varchar	YES
□	contact	varchar	YES

Items per page: 10 1–10 of 17 items 1 of 2 pages

Figure 8-26 Verifying the data in the schema: Columns

2. To view a sample of the data load, the data engineer clicks the **Data sample** tab, as shown in Figure 8-27 on page 119.



Catalog: catalog_01 | **Schema:** marketing_campaign_data

age	job	marital	education	default	balance	housing	loan	contact	day
58	management	married	tertiary	no	2143	yes	no	unknown	5
44	technician	single	secondary	no	29	yes	no	unknown	5
33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
33	unknown	single	unknown	no	1	no	no	unknown	5
35	management	married	tertiary	no	231	yes	no	unknown	5
28	management	single	tertiary	no	447	yes	yes	unknown	5
42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5

Figure 8-27 Verifying the data in the schema: Data sample

8.3 Connecting to and accessing data

After the direct marketing campaign data is ingested into IBM watsonx.data, the data engineer must make the data source available in IBM watsonx.ai so that the team can connect to and access this data for analysis. In this section, the data engineer accesses the IBM watsonx.ai project that the team is using for this initiative and creates a data connection to the IBM watsonx.data instance. With this connection, the team members on the project can access the necessary data for analysis.

8.3.1 Gathering the required information in IBM watsonx.data

Before the data engineer can create a data connection in the IBM watsonx.ai project, they must collect the required information from the IBM watsonx.data instance. The data engineer knows that there are several pieces of information that are needed to establish the connection in IBM watsonx.ai, so they choose to collect the information before creating the data connection.

There are two key locations in IBM watsonx.data:

- Infrastructure manager for the Presto engine connection details
- The Configuration window for the overall connection details

Presto engine-specific connection details

The data engineer gathers the necessary connection information for the Presto engine on IBM watsonx.data by completing the following steps:

1. They navigate to the Infrastructure manager, and click presto-01, the Presto engine, as shown in Figure 8-28.

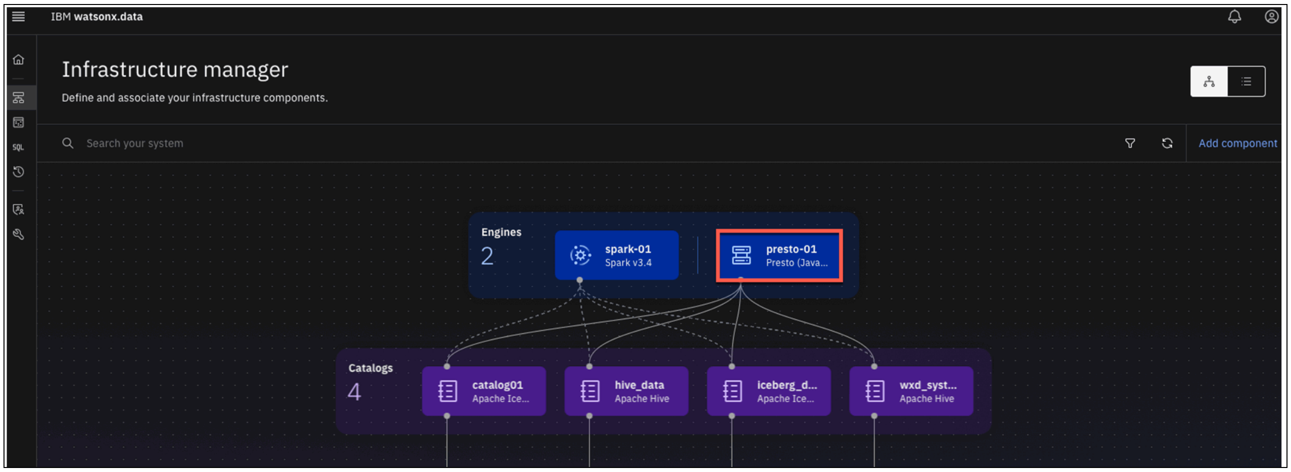
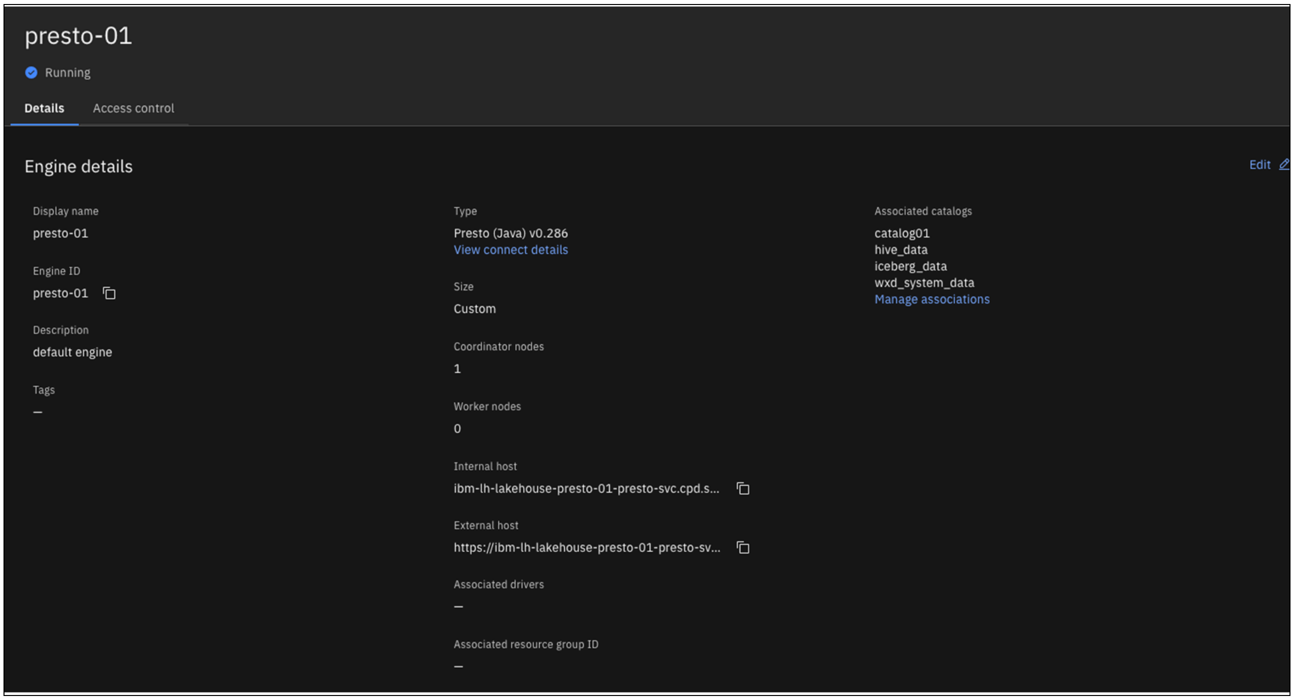


Figure 8-28 Infrastructure manager: presto-01

2. They note the Internal host that is specified in the **Details** tab. The internal host for this Presto engine is as follows:

```
ibm-lh-lakehouse-presto-01-presto-svc.cpd.svc.cluster.local:8443
```

8443 represents the port. The data engineer also notes the engine ID, which in this case is presto-01. This information is used later in IBM watsonx.ai when setting up the connection to the Presto engine on IBM watsonx.data. Figure 8-29 shows the Details tab.



presto-01

Running

Details | Access control

Engine details

Edit

Display name
presto-01

Engine ID
presto-01

Description
default engine

Tags
–

Type
Presto (Java) v0.286
View connect details

Size
Custom

Coordinator nodes
1

Worker nodes
0

Internal host
ibm-lh-lakehouse-presto-01-presto-svc.cpd.s...

External host
https://ibm-lh-lakehouse-presto-01-presto-sv...

Associated drivers
–

Associated resource group ID
–

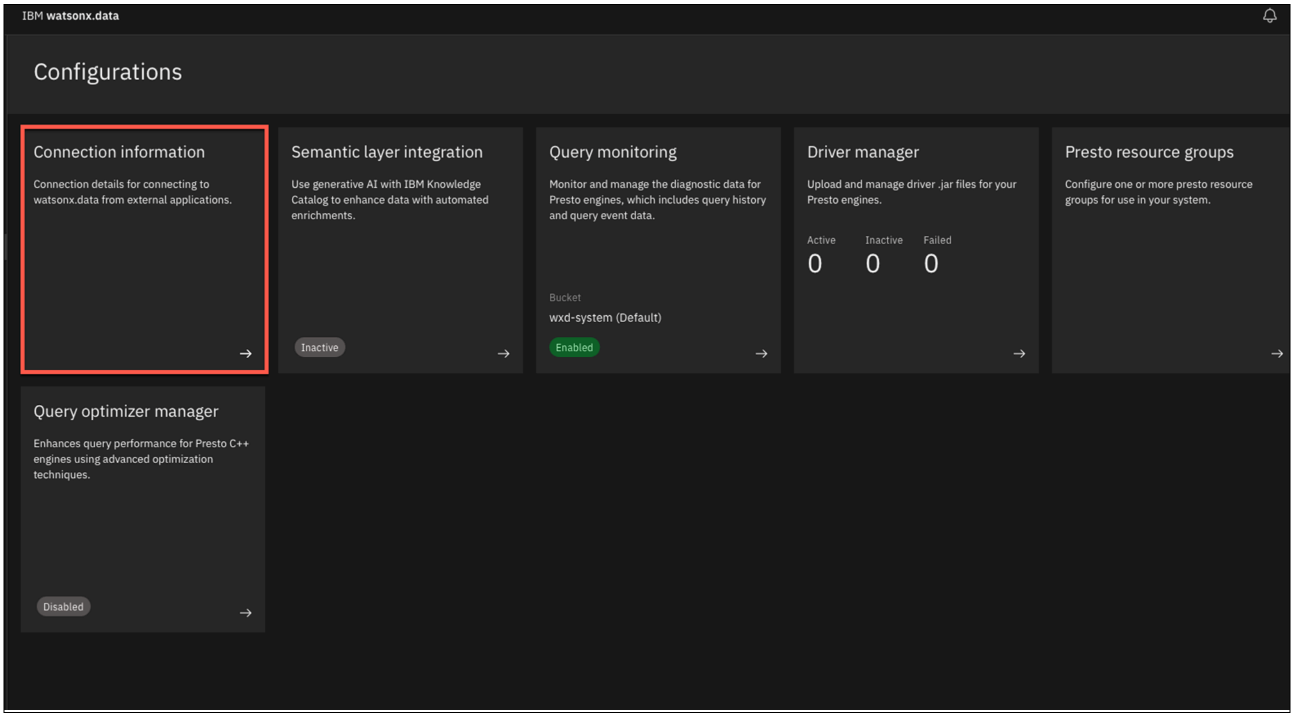
Associated catalogs
catalog01
hive_data
iceberg_data
wxd_system_data
Manage associations

Figure 8-29 Details tab

Overall IBM watsonx.data connection details

The data engineer proceeds to collect additional information that is needed to connect to IBM watsonx.data by using the **Configurations** page. They complete the following steps:

1. On the Configurations page, they click **Connection information**, as shown in Figure 8-30.



IBM watsonx.data

Configurations

Connection information

Connection details for connecting to watsonx.data from external applications.

Semantic layer integration

Use generative AI with IBM Knowledge Catalog to enhance data with automated enrichments.

Inactive

Query monitoring

Monitor and manage the diagnostic data for Presto engines, which includes query history and query event data.

Bucket

wxd-system (Default)

Enabled

Driver manager

Upload and manage driver .jar files for your Presto engines.

Active	Inactive	Failed
0	0	0

Presto resource groups

Configure one or more presto resource groups for use in your system.

Query optimizer manager

Enhances query performance for Presto C++ engines using advanced optimization techniques.

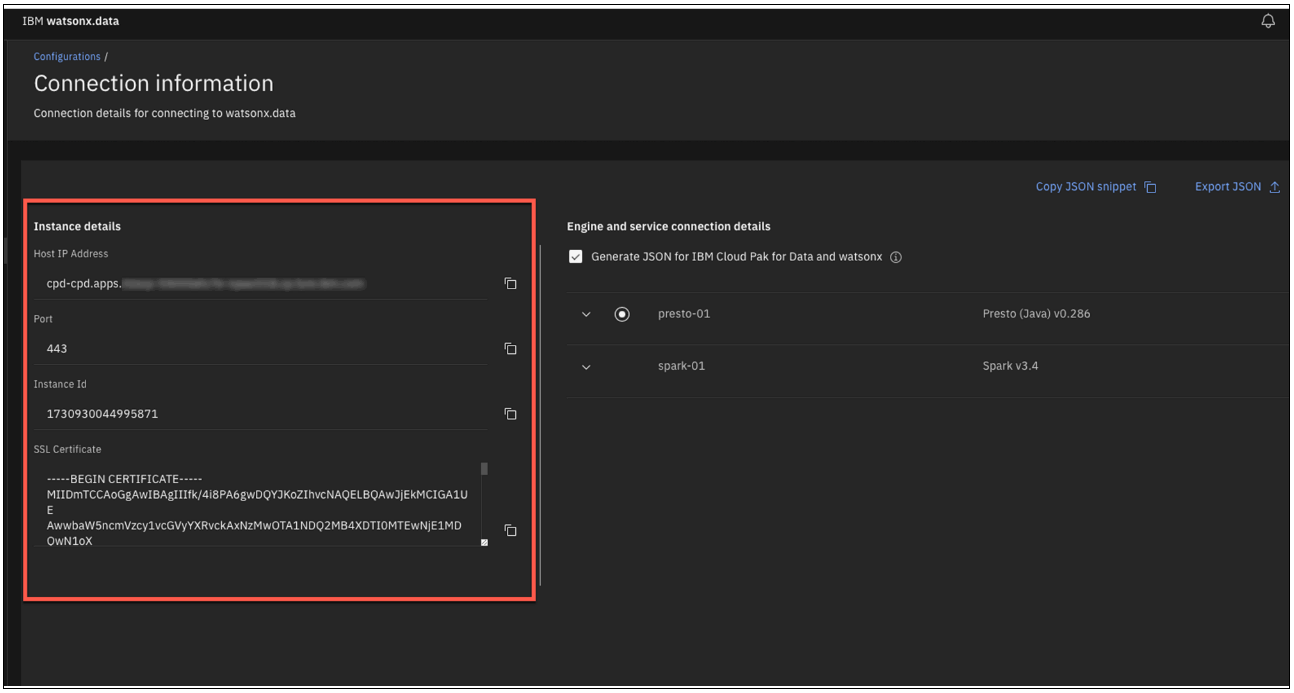
Disabled

Figure 8-30 Connection information

2. From this page, the data engineer notes the values in the following fields:
   - Host IP address
   - Port
   - Instance ID
   - SSL Certificate

This information will be needed later when establishing the connection to IBM watsonx.data from the IBM watsonx.ai project.

Figure 8-31 shows the connection details.



Connection information

Connection details for connecting to watsonx.data

Instance details

Host IP Address: cpd-cpd.apps.[redacted]

Port: 443

Instance Id: 1730930044995871

SSL Certificate:

```
-----BEGIN CERTIFICATE-----
MIIDmTCCAoGgAwIBAgIIIfk/4i8PA6gwDQYJKoZIhvcNAQELBQAwJjEkMCIGA1U
E
AwwbaW5ncmVzcy1vcGVyYXRvckAxNzMwOTA1NDQ2MB4XDTI0MTEwNjE1MD
QwN1oX
```

Engine and service connection details

☑ Generate JSON for IBM Cloud Pak for Data and watsonx

	●	presto-01	Presto (Java) v0.286
		spark-01	Spark v3.4

Figure 8-31 Connection information: Connection details

8.3.2 Creating a data connection in the IBM watsonx.ai project

The data engineer accesses the IBM watsonx.ai project and creates a data connection to the Presto engine in IBM watsonx.data by completing the following steps:

1. From the IBM watsonx.ai home page, the data engineer selects **Projects** → **All projects**, as shown in Figure 8-32.

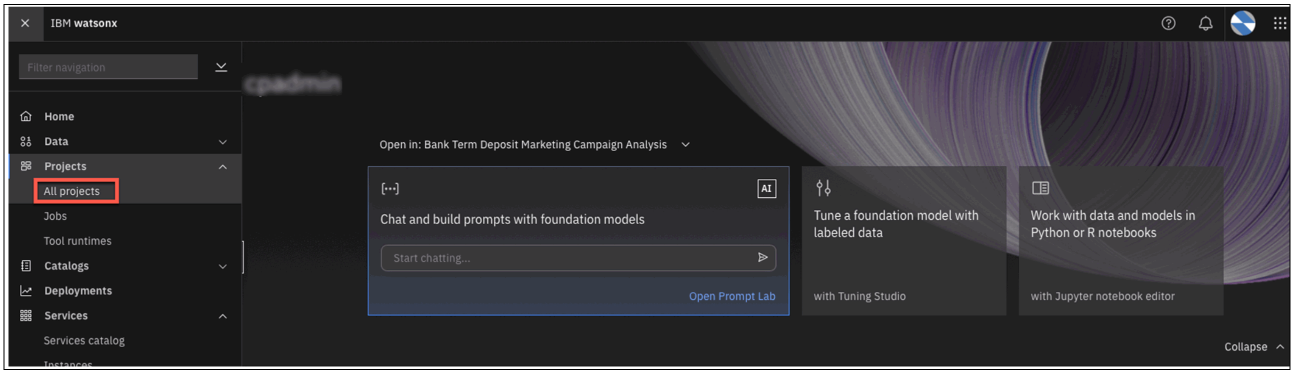
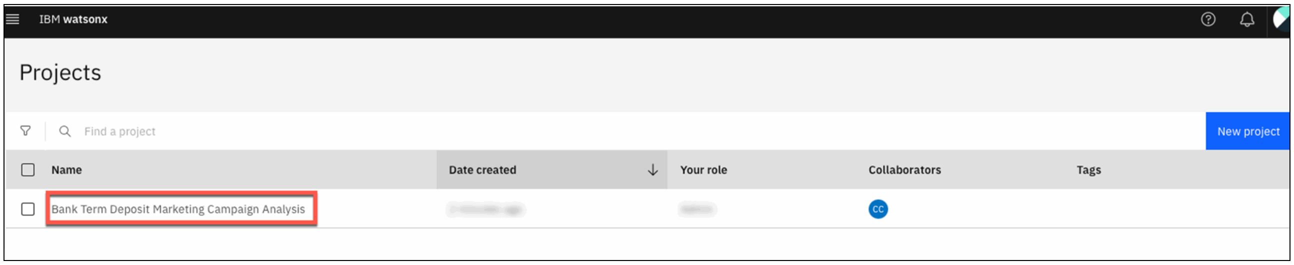

IBM watsonx
Filter navigation
[redacted]
Home
Data
Projects
All projects
Jobs
Tool runtimes
Catalogs
Deployments
Services
Services catalog
Instances
Open in: Bank Term Deposit Marketing Campaign Analysis
AI
Chat and build prompts with foundation models
Start chatting...
Open Prompt Lab
Tune a foundation model with labeled data
with Tuning Studio
Work with data and models in Python or R notebooks
with Jupyter notebook editor
Collapse

Figure 8-32 All projects

2. From the list of projects, they click the Bank Term Deposit Marketing Campaign Analysis project, as shown in Figure 8-33 on page 123.

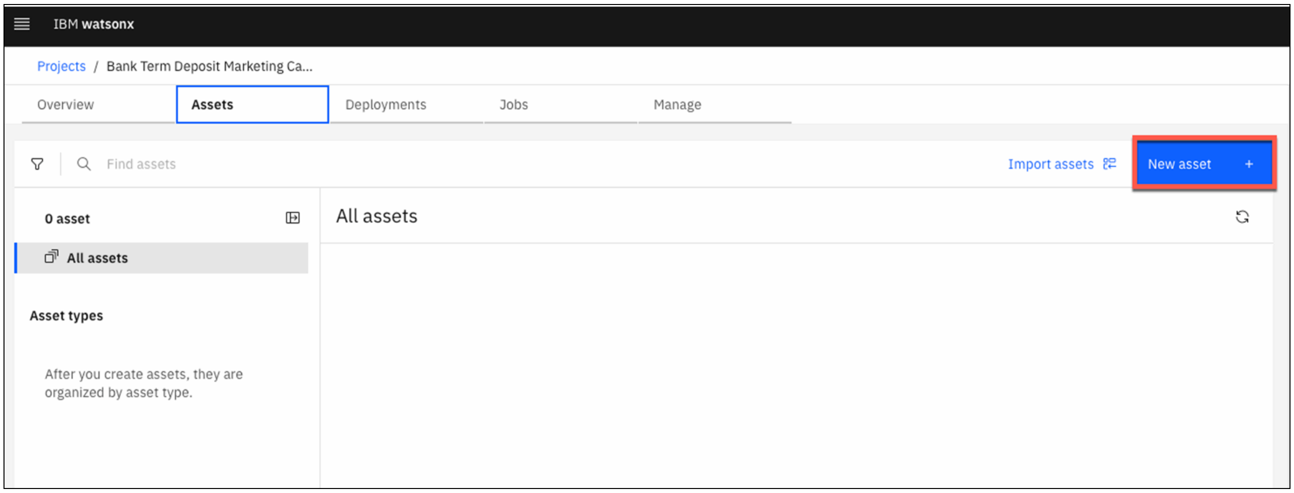


Projects

Name	Date created	Your role	Collaborators	Tags
Bank Term Deposit Marketing Campaign Analysis	[redacted]	[redacted]	CC	

Figure 8-33   Projects - Bank Term Deposit Marketing Campaign Analysis

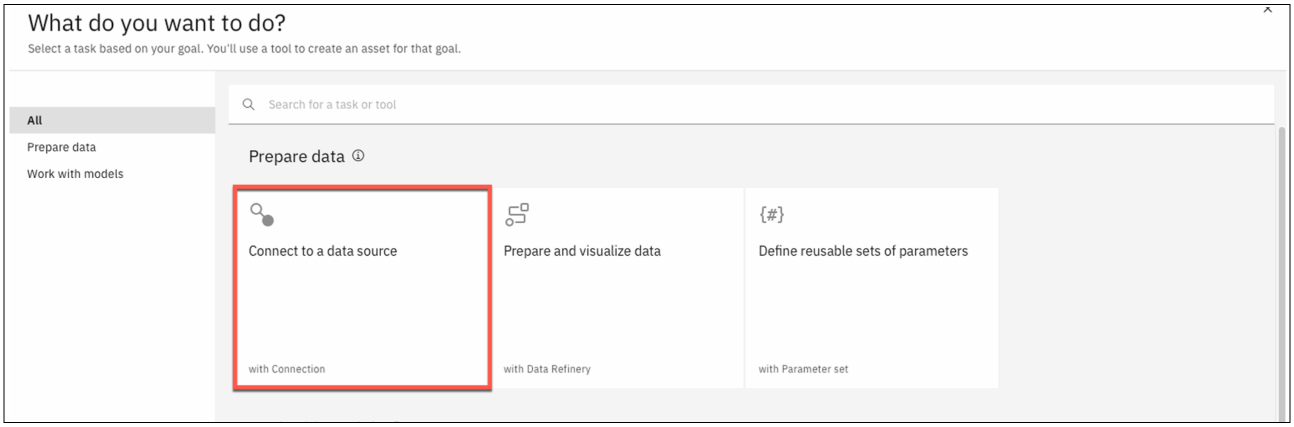
3. To create a data connection, the data engineer navigates to the **Assets** tab in the project and clicks **New asset**, as shown in Figure 8-34.



After you create assets, they are organized by asset type.

Figure 8-34   New asset

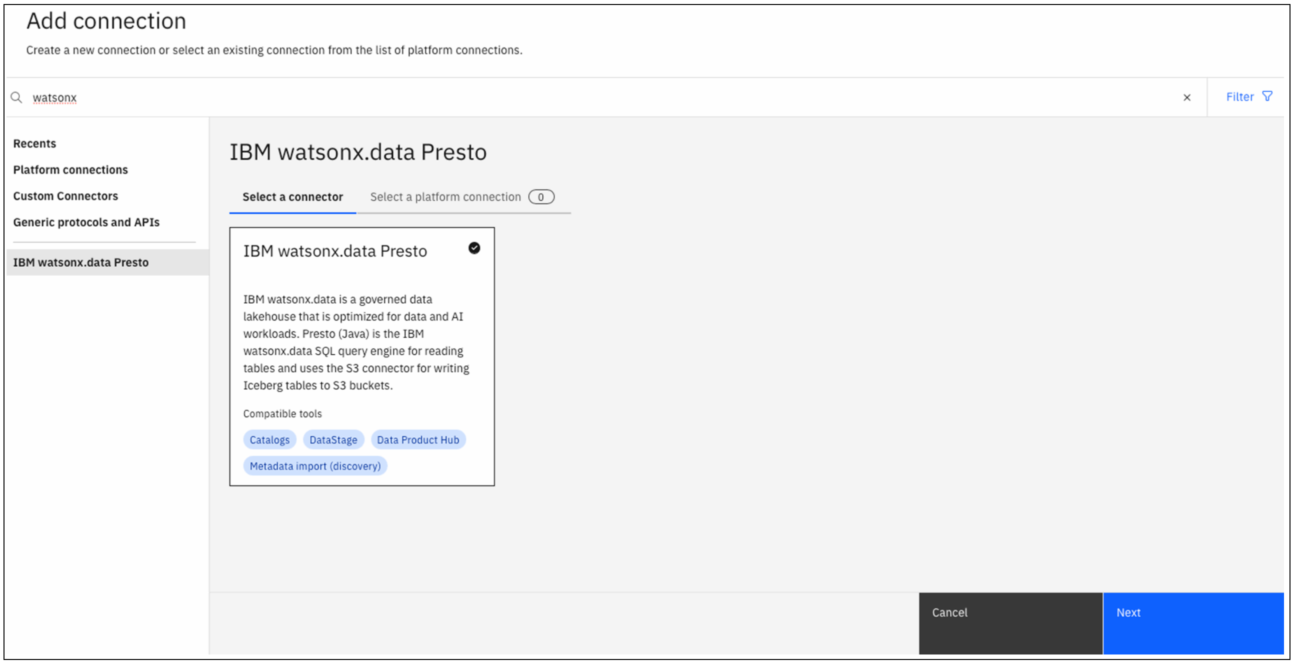
4. Under Prepare data, they click **Connect to a data source**, as shown in Figure 8-35.


What do you want to do?

Select a task based on your goal. You'll use a tool to create an asset for that goal.

Figure 8-35   Connecting to a data source

5. The data engineer wants to establish a connection to the Presto engine on IBM watsonx.data. They search for the IBM watsonx.data Presto connector, and click **Next**, as shown in Figure 8-36.



Add connection

Create a new connection or select an existing connection from the list of platform connections.

watsonx

IBM watsonx.data Presto

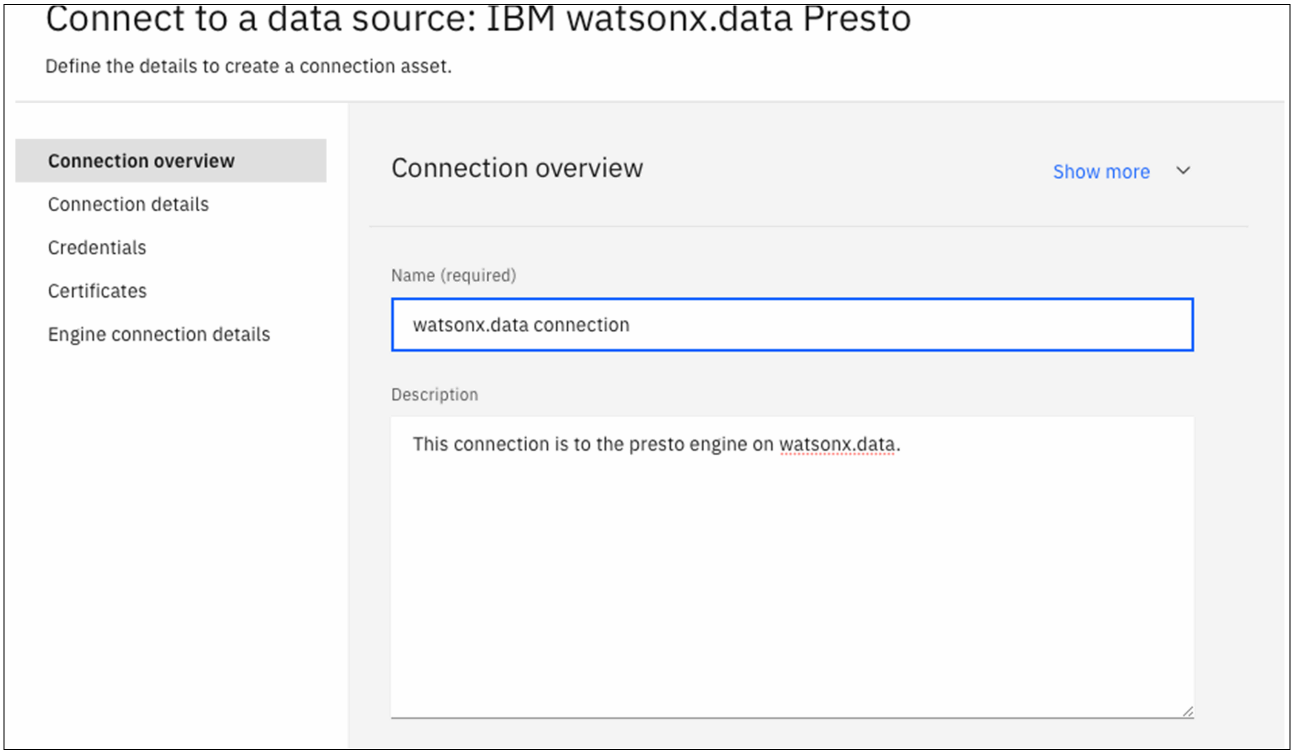
IBM watsonx.data Presto

IBM watsonx.data is a governed data lakehouse that is optimized for data and AI workloads. Presto (Java) is the IBM watsonx.data SQL query engine for reading tables and uses the S3 connector for writing Iceberg tables to S3 buckets.

Compatible tools

Figure 8-36 IBM watsonx.data Presto

6. The data engineer enters the configuration details, starting with the Connection overview window, as shown in Figure 8-37. Within the Connection overview window, they define the connection name and add a description.



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

Connection overview

Name (required)

watsonx.data connection

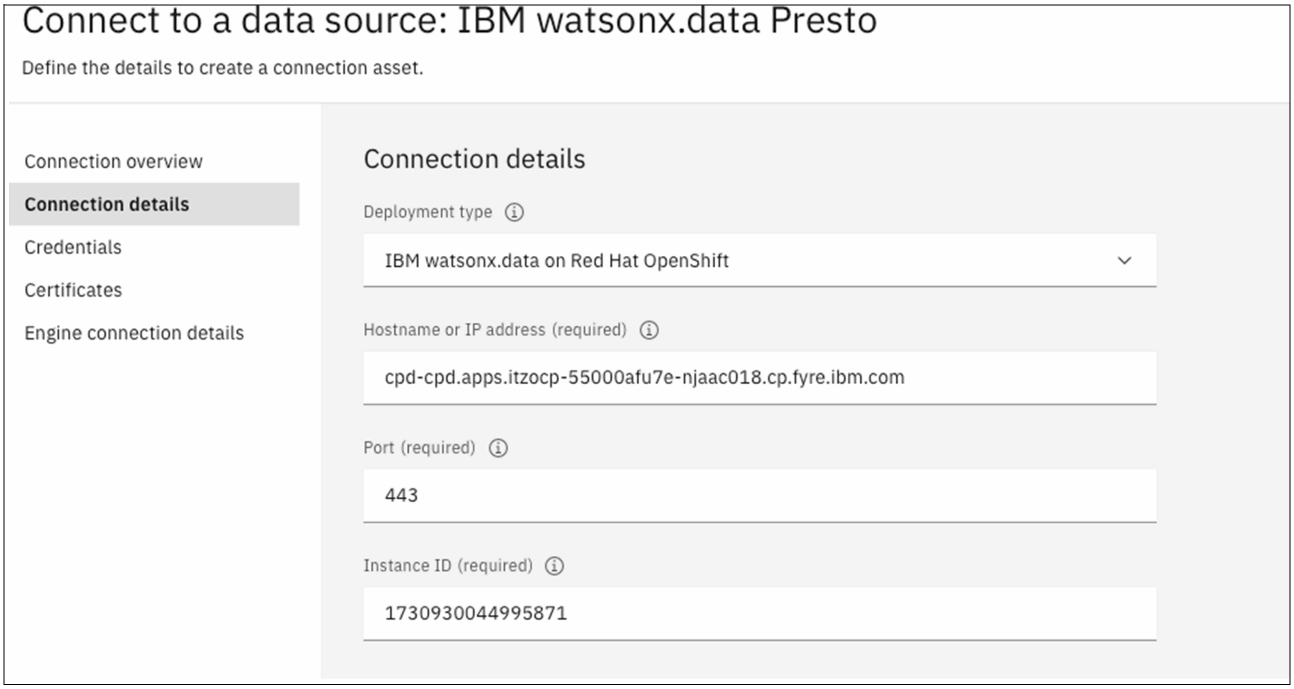
Description

This connection is to the presto engine on watsonx.data.

Figure 8-37 Connection overview

Under Connection details, they update the following fields, as shown in Figure 8-38.

- – Deployment type: IBM watsonx.data on Red Hat OpenShift.
- – Hostname or IP address: This value is retrieved from the Connection information window in IBM watsonx.data.
- – Port: This value is retrieved from the Connection information window in IBM watsonx.data
- – Instance ID: This value is retrieved from the Connection information window in IBM watsonx.data.



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

Connection details

Deployment type: IBM watsonx.data on Red Hat OpenShift

Hostname or IP address (required): cpd-cpd.apps.itzocp-55000afu7e-njaac018.cp.fyre.ibm.com

Port (required): 443

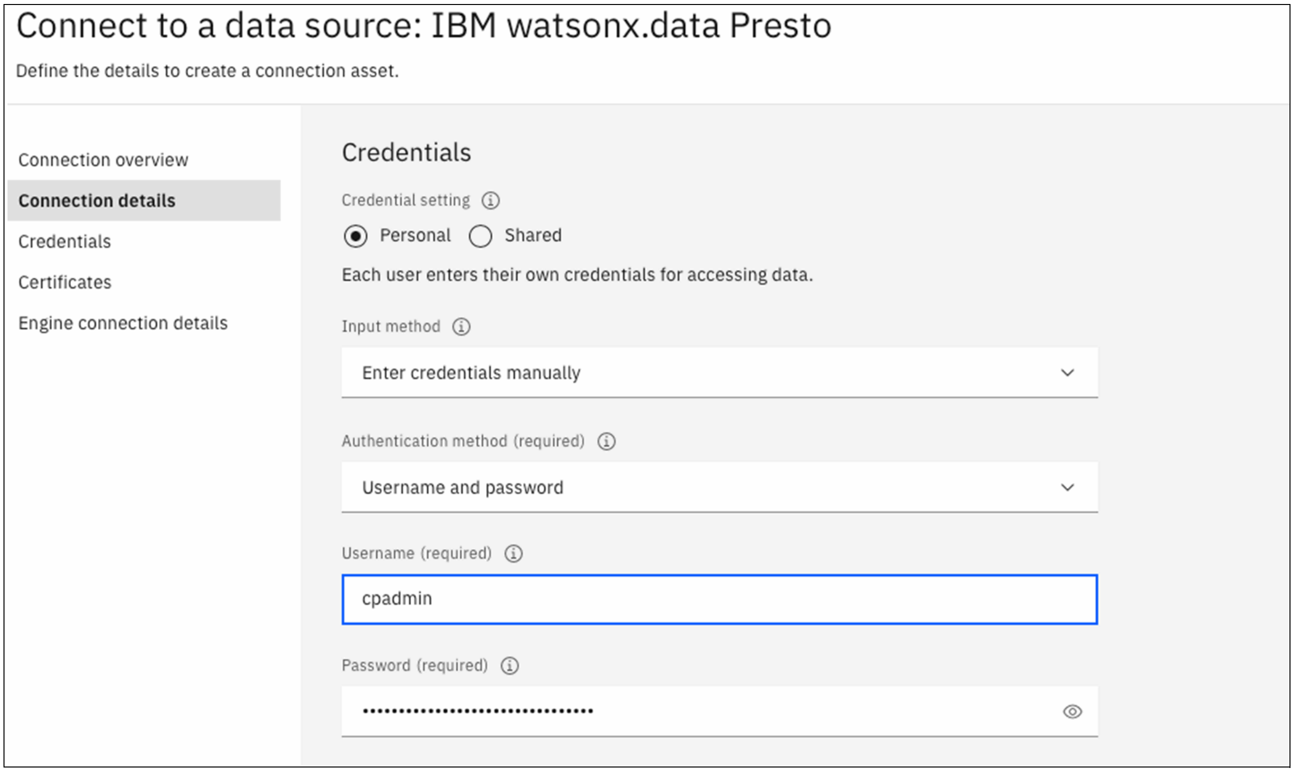
Instance ID (required): 1730930044995871

Figure 8-38 Connection details

7. Under Credentials, the data engineer updates the following fields, as shown in Figure 8-39.

> **Note:** The username and password values vary depending on the user's login credentials

   - Credential setting: Personal
   - Authentication method: Username and password
   - Username: cpadmin
   - Password: password associated with username



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

Credentials

Credential setting

● Personal ○ Shared

Each user enters their own credentials for accessing data.

Input method: Enter credentials manually

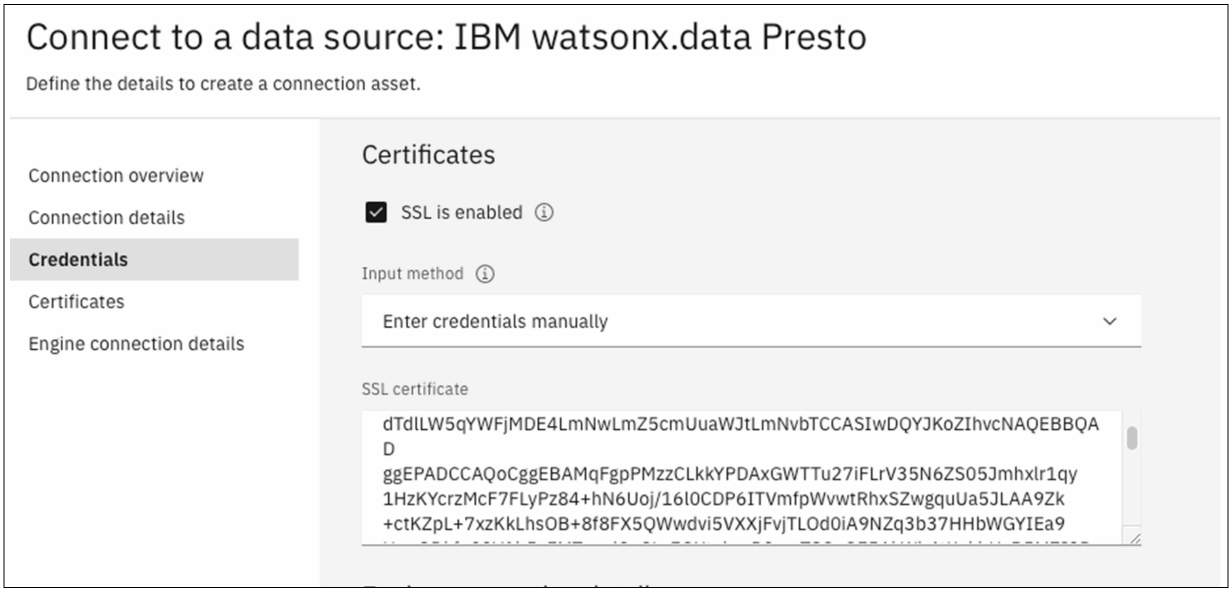
Authentication method (required): Username and password

Username (required): cpadmin

Password (required): ••••••••••••••••••••••••••••••••

Figure 8-39 Credentials

8. Under Certificate, the data engineer ensures that SSL is enabled, and then inputs the SSL certificate that is provided in their IBM watsonx.data instance, as shown in Figure 8-40.



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

Certificates

☑ SSL is enabled

Input method: Enter credentials manually

SSL certificate

```
dTdlLW5qYWFjMDE4LmNwLmZ5cmUuaWJtLmNvbTCCASIwDQYJKoZIhvcNAQEBBQA
D
ggEPADCCAQoCggEBAMqFgpPMzzCLkkYPDAxGWTTu27iFLrV35N6ZS05Jmhxlr1qy
1HzKYcrzMcF7FLyPz84+hN6Uoj/16l0CDP6ITVmfpWvwtRhxSZwgquUa5JLAA9Zk
+ctKZpL+7xzKkLhsOB+8f8FX5QWwdvi5VXXjFvjTLOd0iA9NZq3b37HHbWGYIEa9
```

Figure 8-40 Certificates

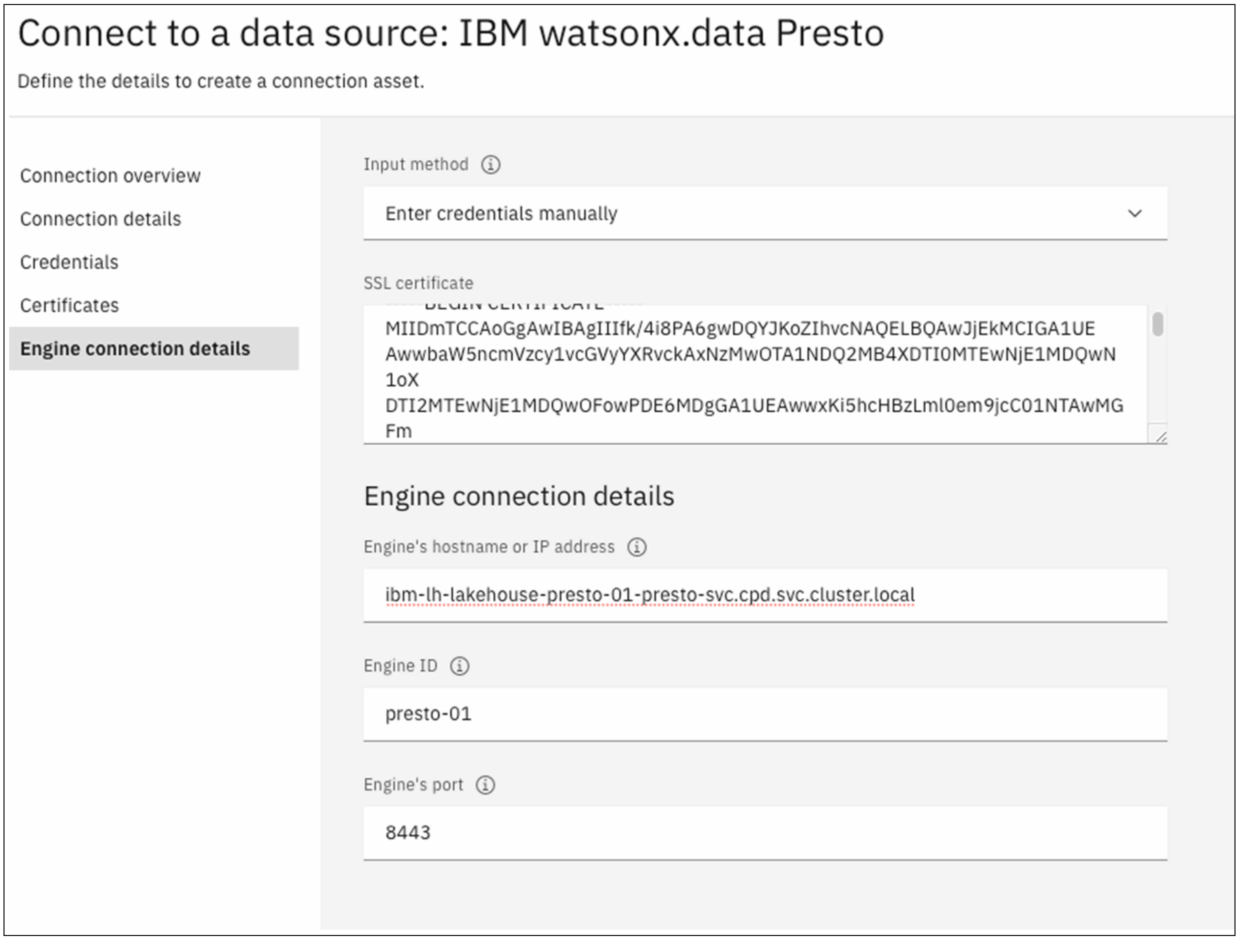
9. The data engineer adds the connection details that are specific to the Presto engine on IBM watsonx.data, as shown in Figure 41:

   – Engine's hostname or IP address:

     `ibm-lh-lakehouse-presto-01-presto-svc.cpd.svc.cluster.local`

   – Engine ID: presto-01

   – Engine's port: 8443



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

Input method: Enter credentials manually

SSL certificate

```
-----BEGIN CERTIFICATE-----
MIIDmTCCAoGgAwIBAgIIIfk/4i8PA6gwDQYJKoZIhvcNAQELBQAwJjEkMCIGA1UE
AwwbaW5ncmVzcy1vcGVyYXRvckAxNzMwOTA1NDQ2MB4XDTI0MTEwNjE1MDQwN
1oX
DTI2MTEwNjE1MDQwOFowPDE6MDgGA1UEAwwxKi5hcHBzLml0em9jcC01NTAwMG
Fm
```

Engine connection details

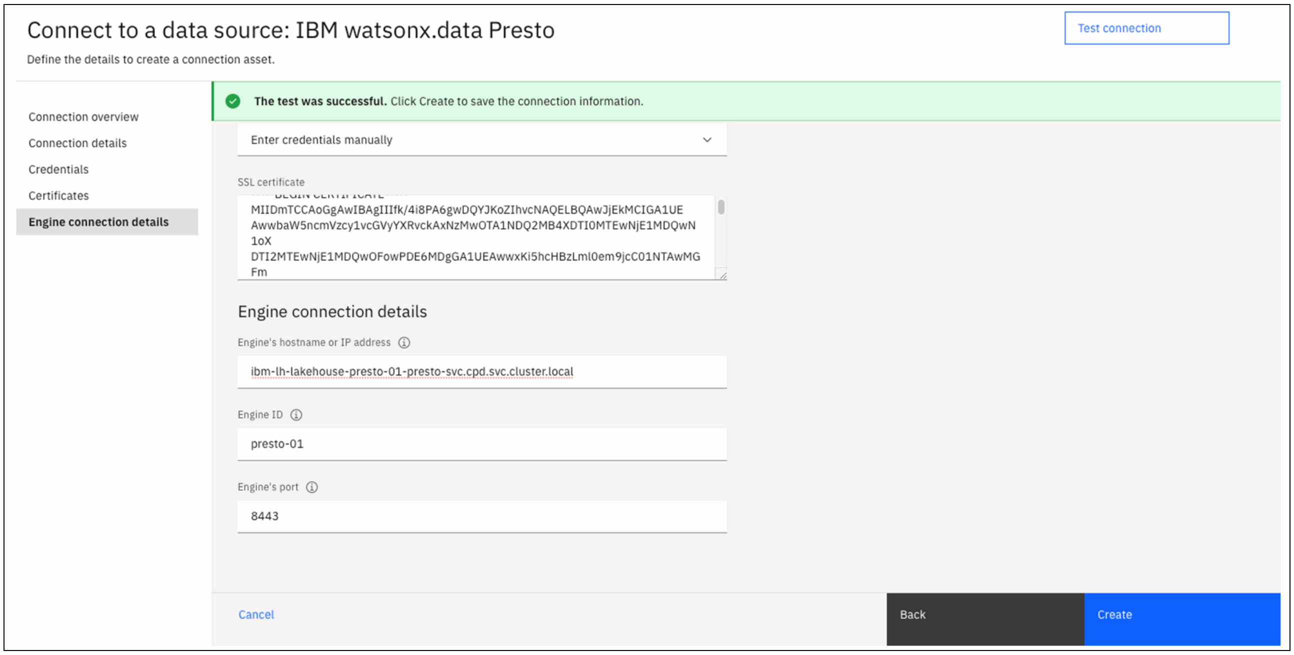
Engine's hostname or IP address: ibm-lh-lakehouse-presto-01-presto-svc.cpd.svc.cluster.local

Engine ID: presto-01

Engine's port: 8443

Figure 8-41 Engine connection details

10. The data engineer tests the connection. Because the test was successful, they click **Create** to create the IBM watsonx.data Presto connection in the project, as shown in Figure 8-42 on page 129.



Connect to a data source: IBM watsonx.data Presto

Define the details to create a connection asset.

The test was successful. Click Create to save the connection information.

Enter credentials manually

SSL certificate

```
-----BEGIN CERTIFICATE-----
MIIDmTCCAoGgAwIBAgIIIfk/4i8PA6gwDQYJKoZIhvcNAQELBQAwJjEkMCIGA1UE
AwwbaW5ncmVzcy1vcGVyYXRvckAxNzMwOTA1NDQ2MB4XDTI0MTEwNjE1MDQwN
1oX
DTI2MTEwNjE1MDQwOFowPDE6MDgGA1UEAwwxKi5hcHBzLml0em9jcC01NTAwMG
Fm
```

Engine connection details

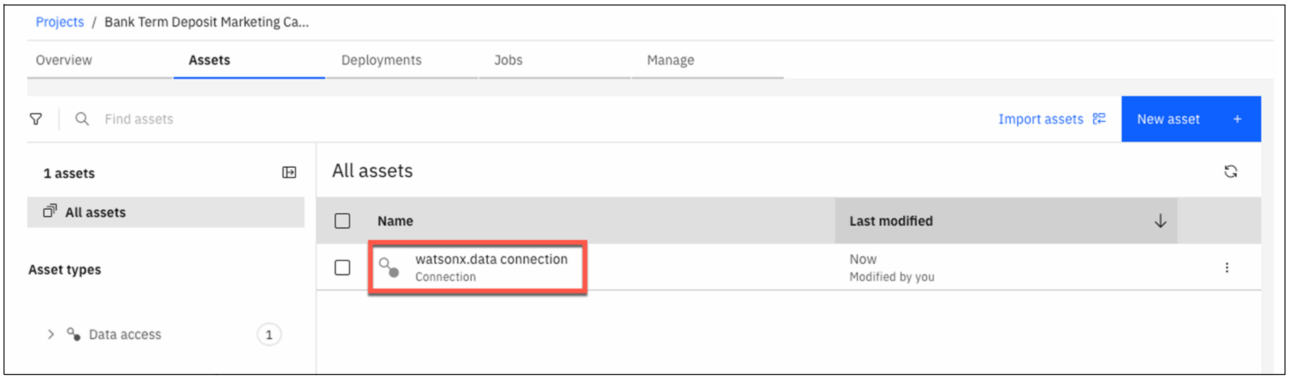
Engine's hostname or IP address: ibm-lh-lakehouse-presto-01-presto-svc.cpd.svc.cluster.local

Engine ID: presto-01

Engine's port: 8443

Figure 8-42 Test connection: Create

11. After the connection is created, the data engineer views the connection in the **Assets** tab of the project by selecting **Projects** → **Bank Term Deposit Marketing Campaign Analysis** → **Assets** → **watsonx.data connection**, as shown in Figure 8-43.



☐	Name	Last modified
☐	watsonx.data connection Connection	Now Modified by you

Figure 8-43 IBM watsonx.data connection

8.4 Visualizing and exploring data

After the data connection to IBM watsonx.data is established in the IBM watsonx.ai project, the data analyst can use the connection. In this section, the data analyst uses the data connection that was created by the data engineer in the IBM watsonx.ai project to modify the data set and generate a data visualization by using the Data Refinery feature.

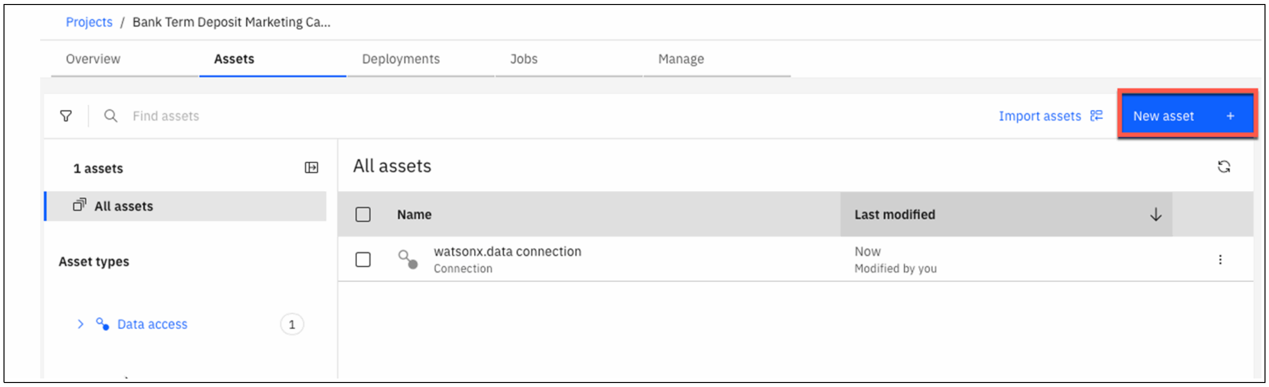
8.4.1 Creating a Data Refinery flow

In this section, the data analyst creates a Data Refinery flow to filter the data set and create a data visualization to visualize the number of bank term deposits by profession.

Creating a Data Refinery flow in an IBM watsonx.ai project

The data analyst starts by creating a Data Refinery asset by completing the following steps:

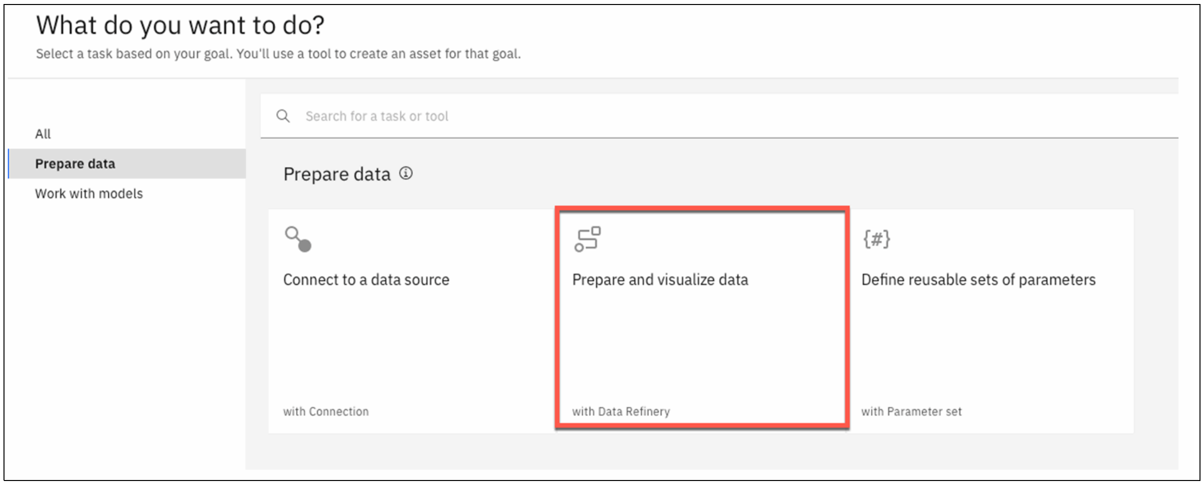
1. From the **Assets** tab in the Bank Term Deposit Marketing Campaign Analysis project, they click **New Asset**, as shown in Figure 8-44.



☐	Name	Last modified
☐	watsonx.data connection Connection	Now Modified by you

Figure 8-44 New asset

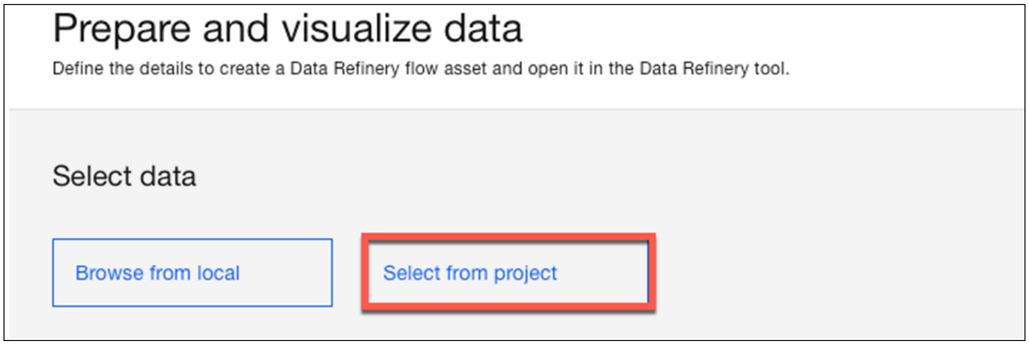
2. Under Prepare data, the data analyst clicks **Prepare and visualize data with Data Refinery**, as shown in Figure 8-45.


What do you want to do?

Select a task based on your goal. You'll use a tool to create an asset for that goal.

Figure 8-45 Prepare and visualize data with Data Refinery

3. They select data for the Data Refinery flow by clicking **Select from project**, as shown in Figure 8-46.


Prepare and visualize data

Define the details to create a Data Refinery flow asset and open it in the Data Refinery tool.

Figure 8-46 Select from project

4. The data that is needed for this analysis comes from the IBM watsonx.data connection that was established by the data engineer earlier. To find the data source that is needed for this analysis, the data analyst selects **Connection** → **watsonx.data connection**, as shown in Figure 8-47.

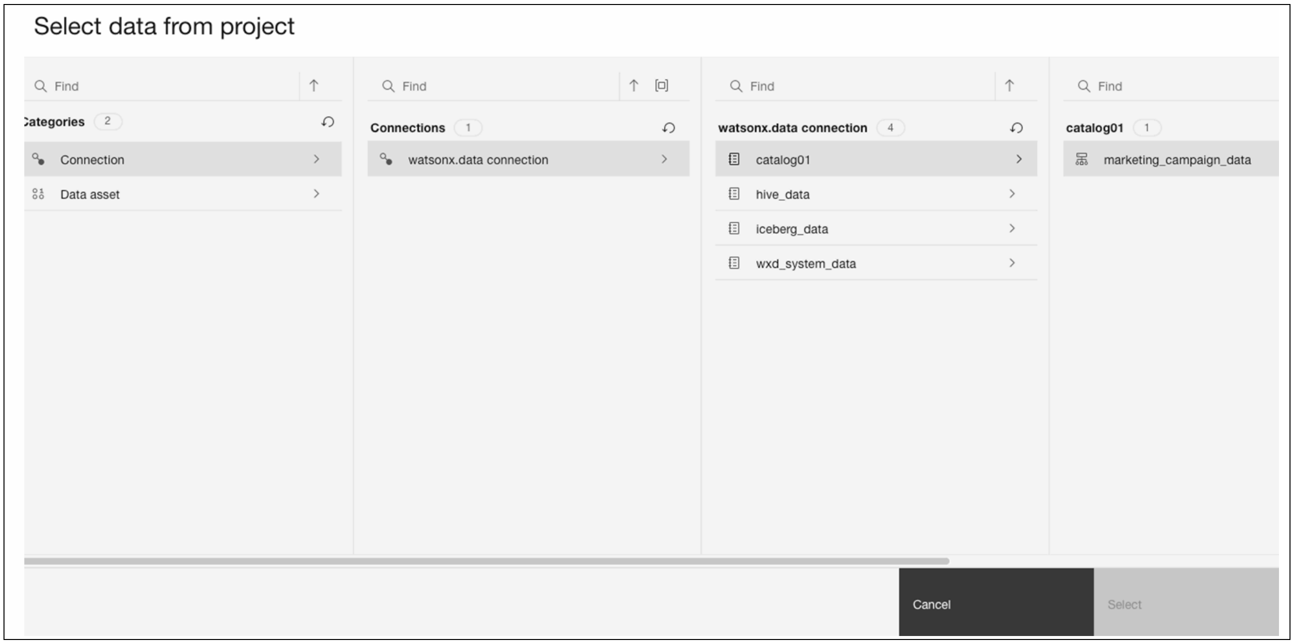
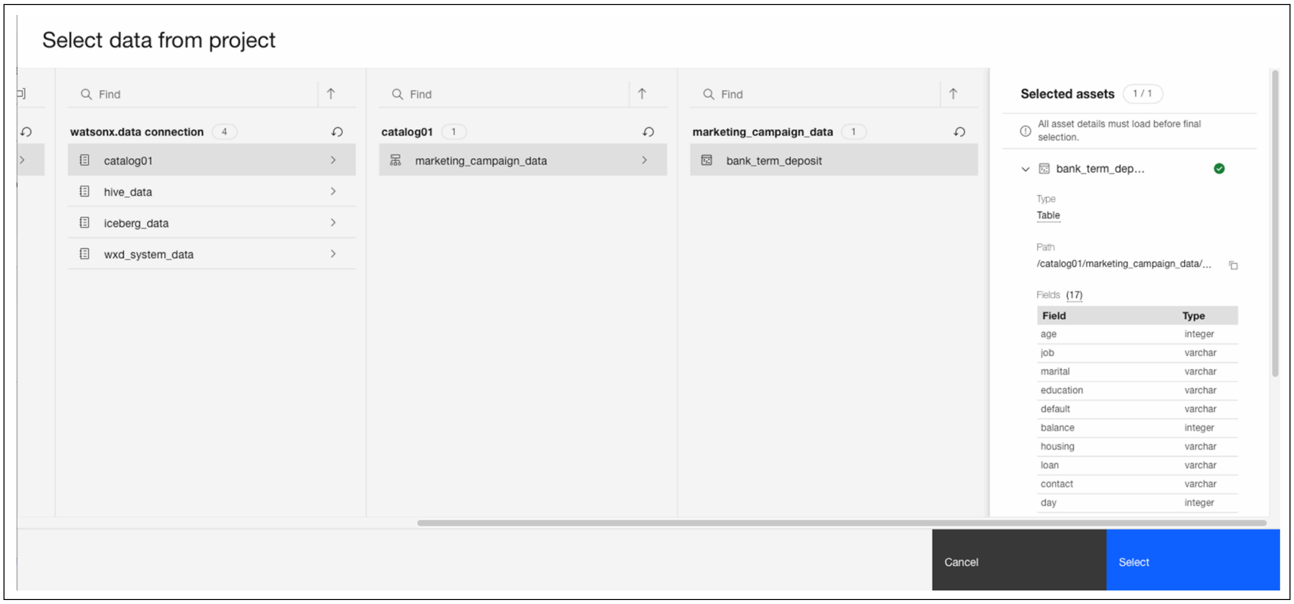


Figure 8-47 Categories: Connections

5. The data analyst selects the corresponding catalog, schema, and table (**watsonx.data connection** → **catalog01** → **marketing_campaign_data** → **bank_term_deposit**), as shown in Figure 8-48.



All asset details must load before final selection.

Type
Table

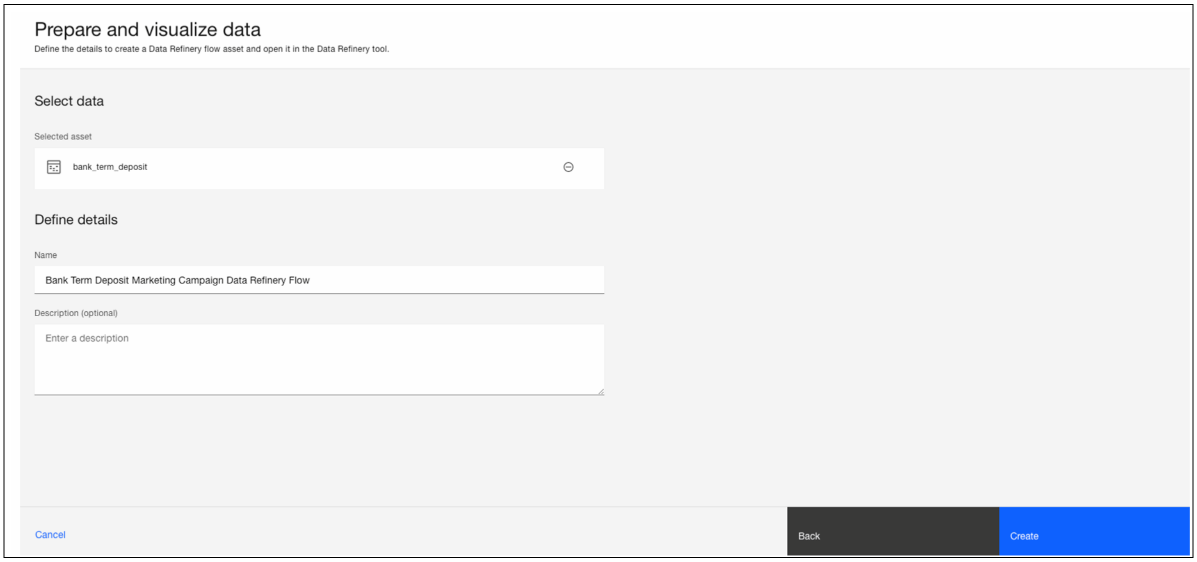
Path
/catalog01/marketing_campaign_data/...

Fields (17)

Field	Type
age	integer
job	varchar
marital	varchar
education	varchar
default	varchar
balance	integer
housing	varchar
loan	varchar
contact	varchar
day	integer

Figure 8-48 Selecting the corresponding catalog, schema, and table

6. Before selecting **Create**, the data analyst enters a name for the Data Refinery flow, as shown in Figure 8-49.



Prepare and visualize data

Define the details to create a Data Refinery flow asset and open it in the Data Refinery tool.

Select data

Selected asset: bank_term_deposit

Define details

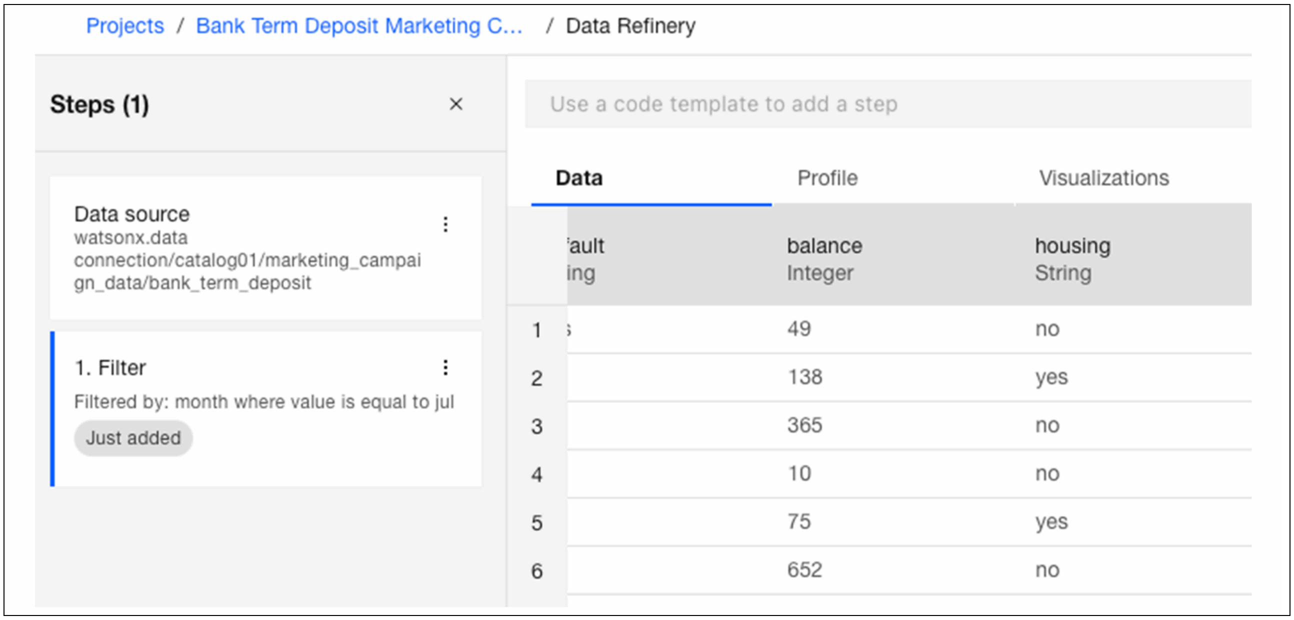
Name: Bank Term Deposit Marketing Campaign Data Refinery Flow

Description (optional): Enter a description

Figure 8-49 Define the details: Create

Filtering data with Data Refinery

Before generating any visualizations, the data analyst filters the data to reflect the period that is needed for the analysis. As shown in Figure 8-50, the data analyst adds a filter where the month is equal to “`jul`” (for July), which filters the data to show only interactions from the direct marketing campaign in the month of July.



Projects / Bank Term Deposit Marketing C... / Data Refinery
Steps (1)
Use a code template to add a step
Data
Profile
Visualizations

Data source
watsonx.data connection/catalog01/marketing_campaign_data/bank_term_deposit

1. Filter
Filtered by: month where value is equal to jul
Just added

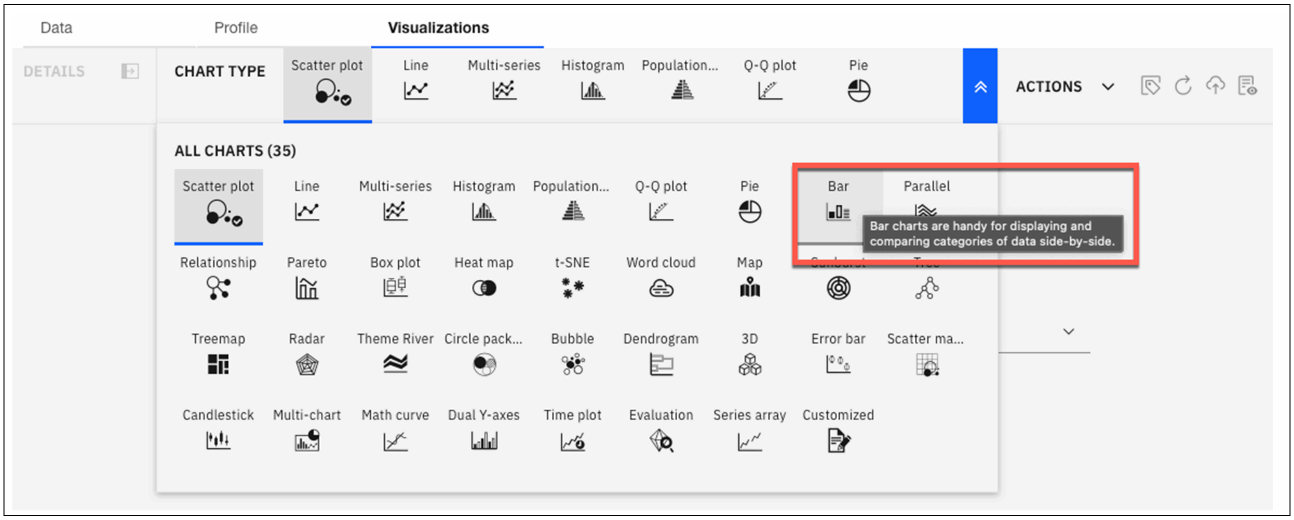
	'ault ing	balance Integer	housing String
1	;	49	no
2		138	yes
3		365	no
4		10	no
5		75	yes
6		652	no

Figure 8-50 Selecting only interactions where the month field equals “jul”

Creating and formatting a data visualization

Now, the data analyst creates and formats a data visualization by completing the following steps:

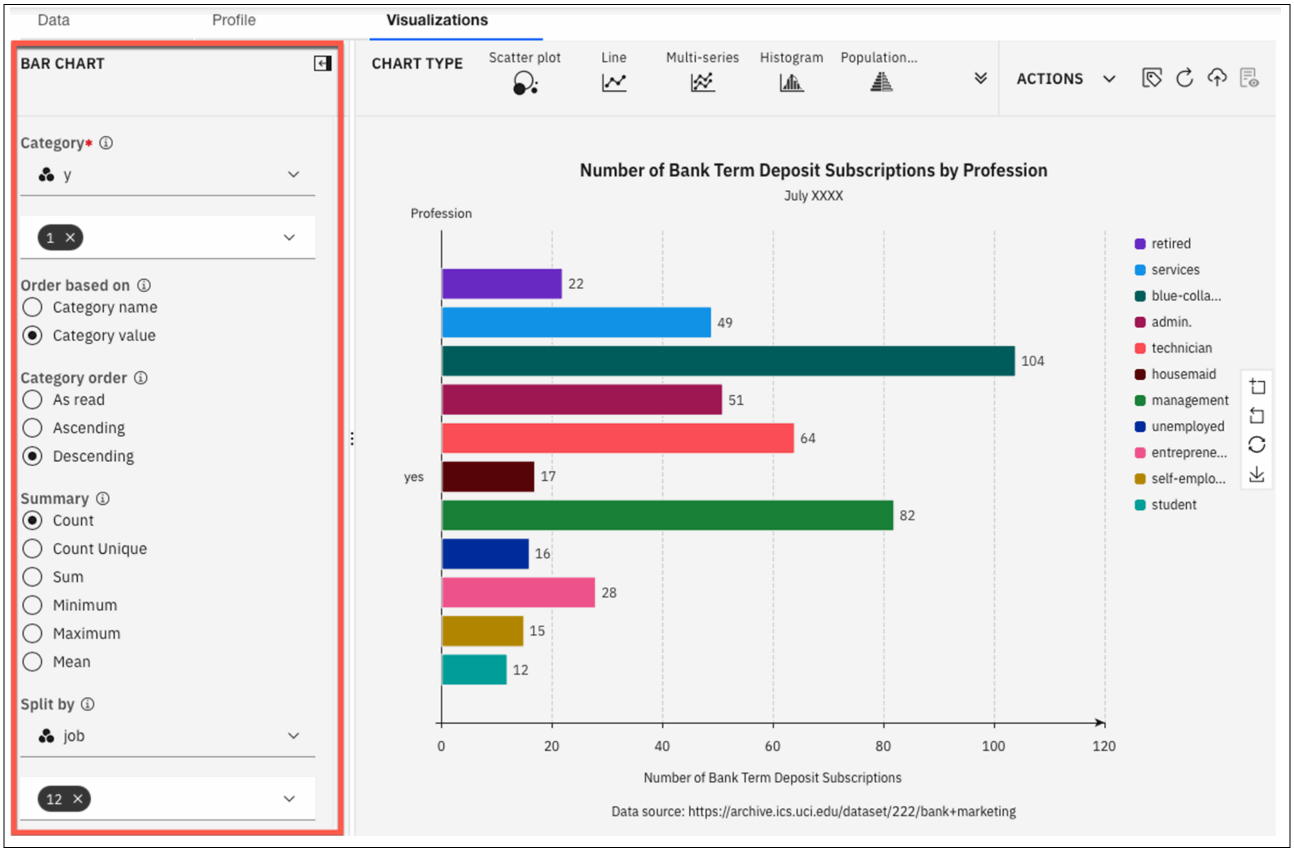
1. After the data is ready for analysis, the data analyst clicks the **Visualizations** tab, and then clicks the **Bar chart type**, as shown in Figure 8-51.



Bar charts are handy for displaying and comparing categories of data side-by-side.

Figure 8-51 Visualizations - Bar

2. As shown in Figure 8-52 and Figure 8-53 on page 135, the data analyst uses the fields on the left to generate a chart that shows the number of bank term deposit subscriptions by profession. Here, the data analyst can modify the categories that appear in the visual, the chart and axis titles, and the type of bar chart (for example, horizontal, vertical, stacked, clustered, and others).



Data | Profile | Visualizations

BAR CHART

Category* y

1 ×

Order based on
○ Category name
● Category value

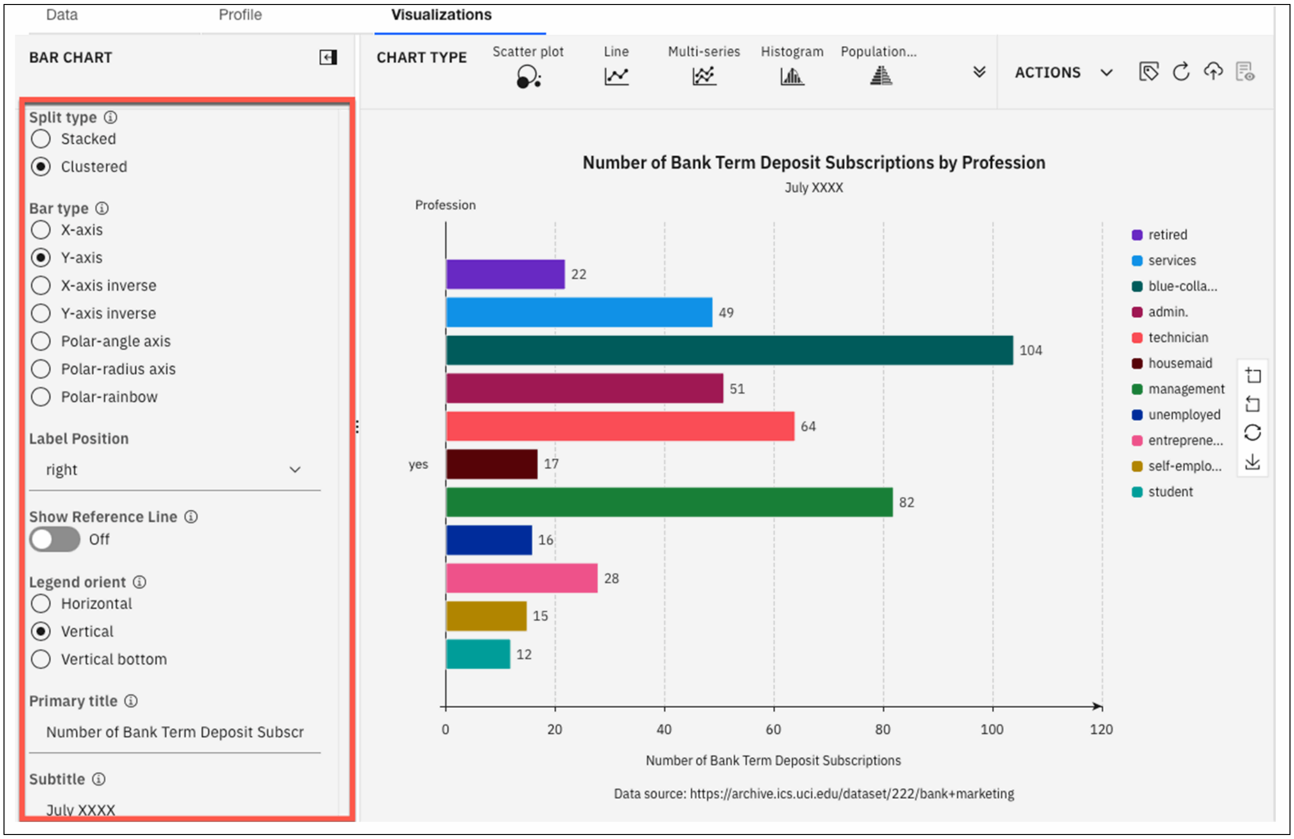
Category order
○ As read
○ Ascending
● Descending

Summary
● Count
○ Count Unique
○ Sum
○ Minimum
○ Maximum
○ Mean

Split by job

12 ×

Figure 8-52 Bar chart configuration window



Split type ⓘ
○ Stacked
● Clustered

Bar type ⓘ
○ X-axis
● Y-axis
○ X-axis inverse
○ Y-axis inverse
○ Polar-angle axis
○ Polar-radius axis
○ Polar-rainbow

Label Position
right

Show Reference Line ⓘ
Off

Legend orient ⓘ
○ Horizontal
● Vertical
○ Vertical bottom

Primary title ⓘ
Number of Bank Term Deposit Subscr

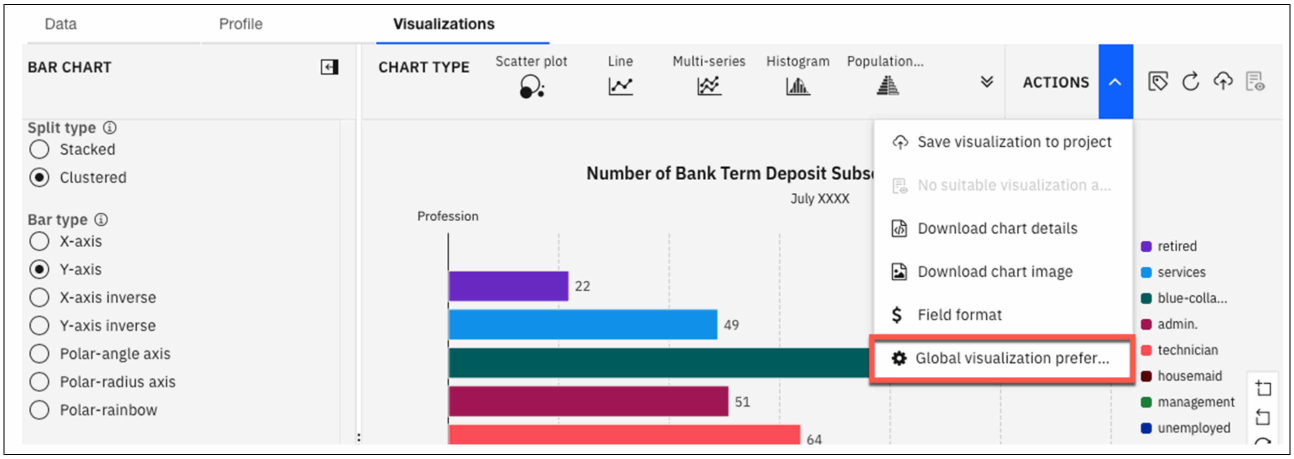
Subtitle ⓘ
July XXXX

Figure 8-53 Bar chart configuration window (continued)

Additional data visualization formatting

The data analyst wants to further format this visualization, specifically the color scheme. The business user requesting this visualization might share the results with others within the organization or in presentations, so the data analyst wants to help ensure that the visualization adheres to the company's color and formatting standards. They data analyst completes the following steps:

1. To modify the color scheme for the data visualization, the data analyst clicks **Actions** selects **Global visualization preferences**, as shown in Figure 8-54.

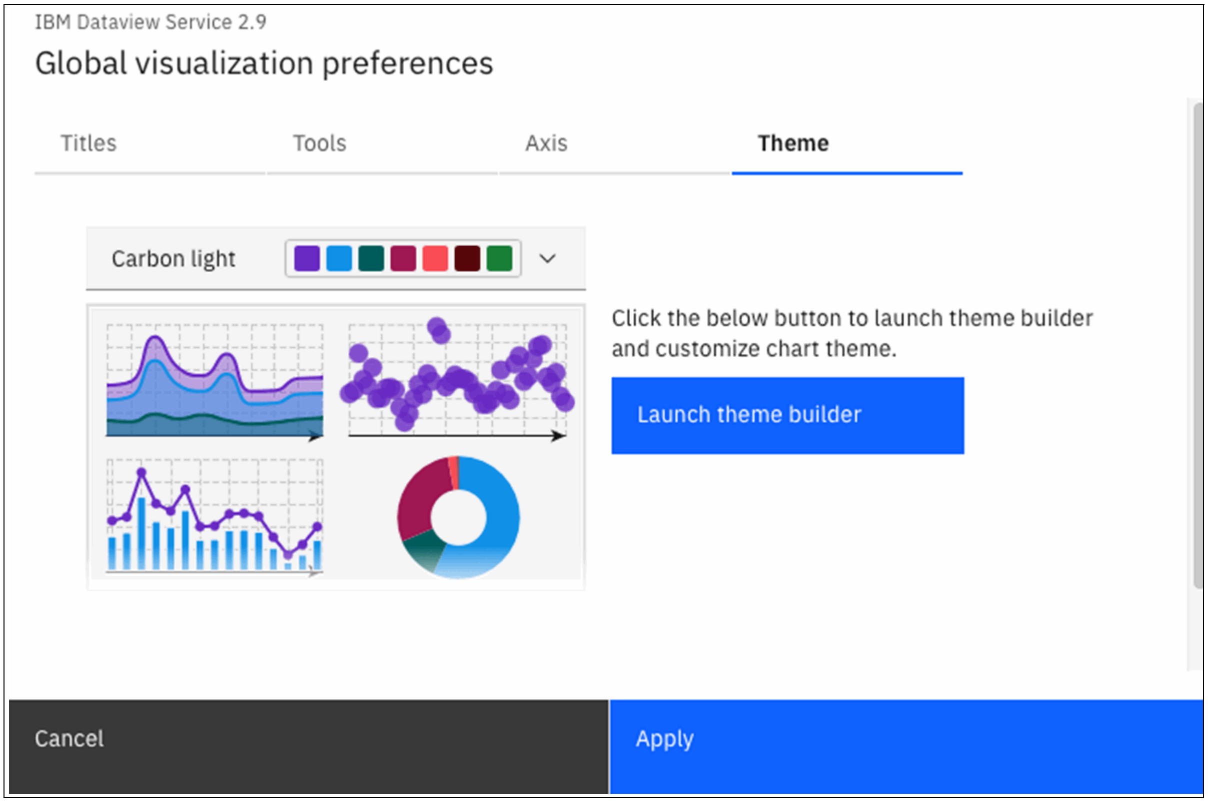


Split type ⓘ
○ Stacked
● Clustered

Bar type ⓘ
○ X-axis
● Y-axis
○ X-axis inverse
○ Y-axis inverse
○ Polar-angle axis
○ Polar-radius axis
○ Polar-rainbow

Figure 8-54 Global visualization preferences

2. Next, click the **Theme** tab. Then click **Launch theme builder**, as shown in Figure 8-55.

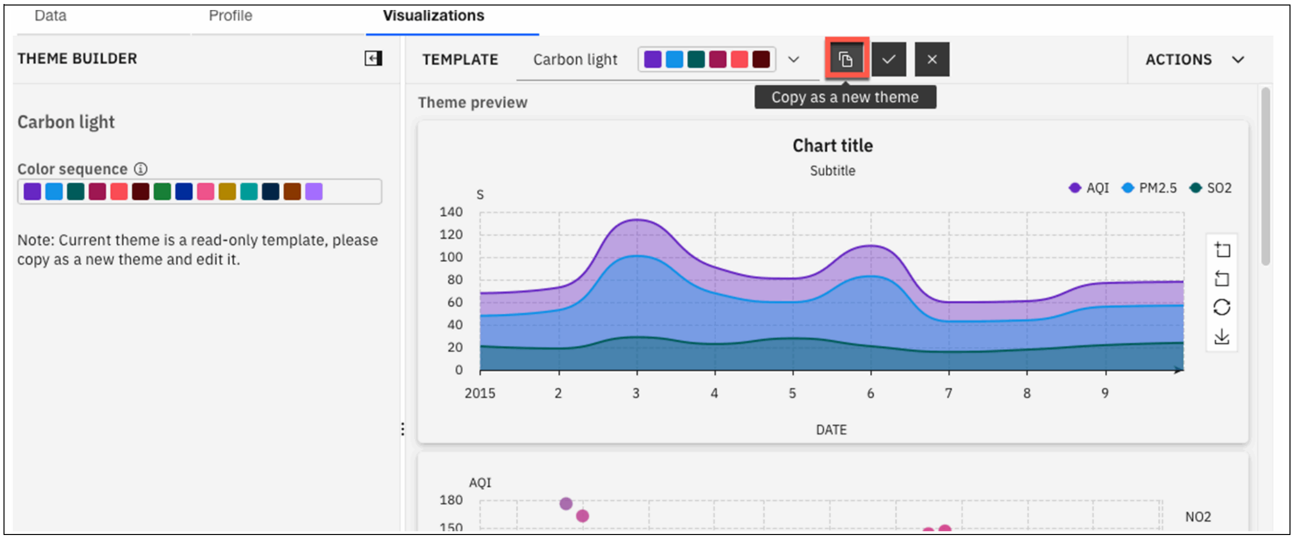

IBM Dataview Service 2.9

Global visualization preferences

Click the below button to launch theme builder and customize chart theme.

Figure 8-55 Launching the theme builder

3. To create a theme, the data analyst clicks **Copy as a new theme**, as shown in Figure 8-56.

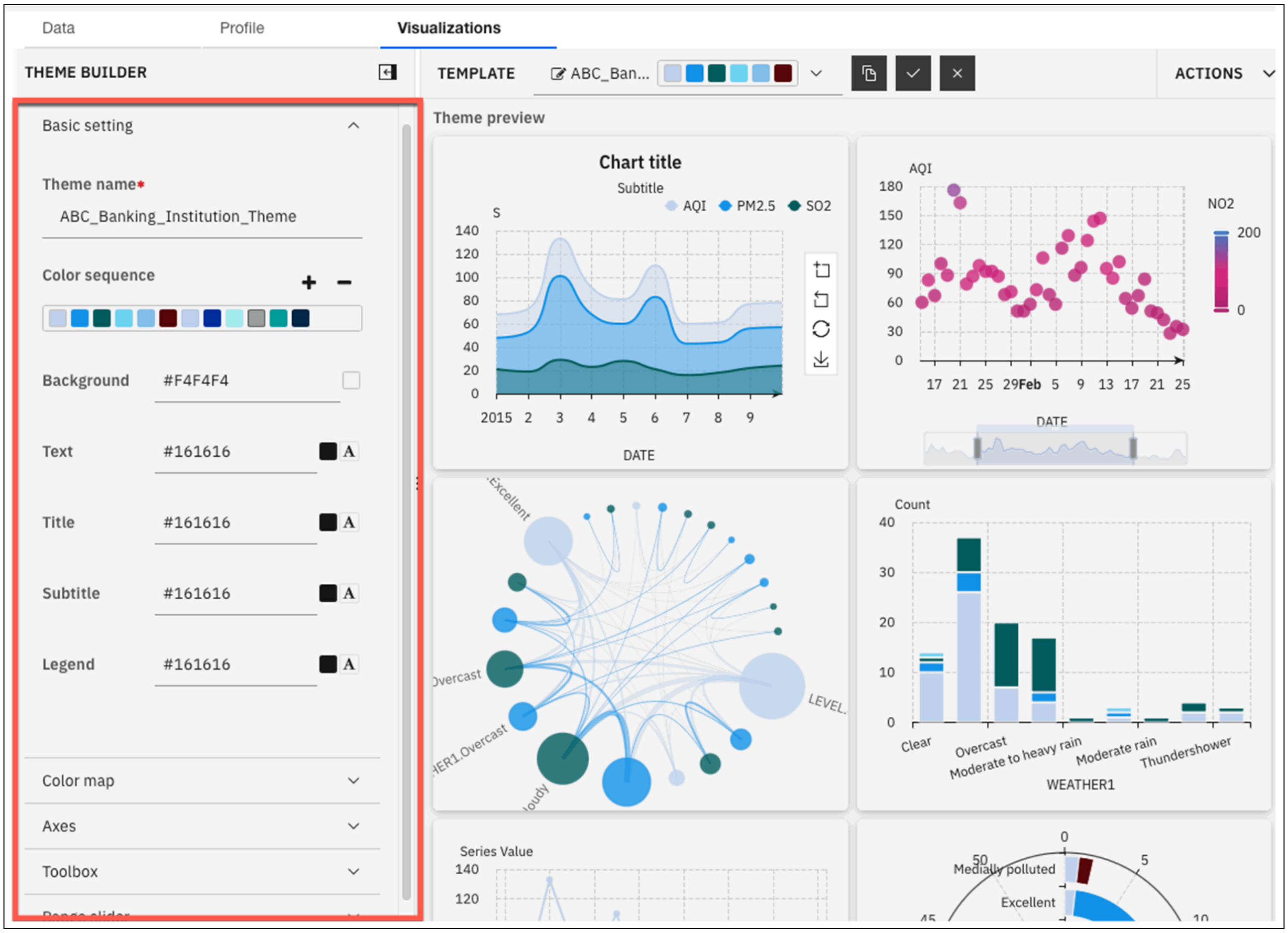


Note: Current theme is a read-only template, please copy as a new theme and edit it.

Figure 8-56 Copying as a new theme

4. Use the fields in the left pane to format the new theme, including the theme name, color sequence, background color, legend and axis formatting, and other attributes.

Figure 8-57 shows the specifications for the new theme that was generated by the data analyst, which is named ABC_Banking_Institution_Theme.



THEME BUILDER

Basic setting

Theme name*: ABC_Banking_Institution_Theme

Color sequence

Background: #F4F4F4

Text: #161616

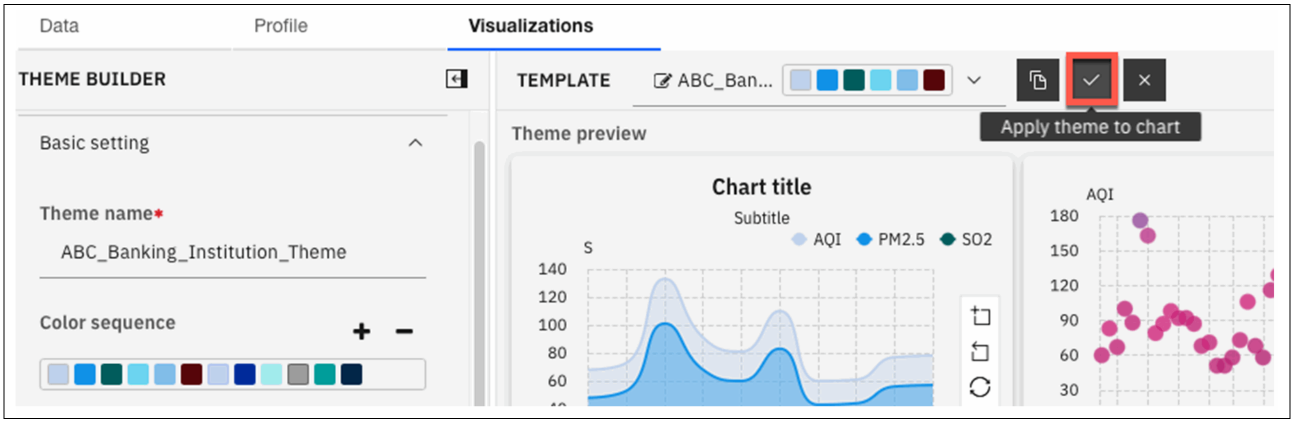
Title: #161616

Subtitle: #161616

Legend: #161616

Figure 8-57 Theme builder settings

5. After the theme is created, the data analyst clicks **Apply theme to chart**, as shown in Figure 8-58.



THEME BUILDER

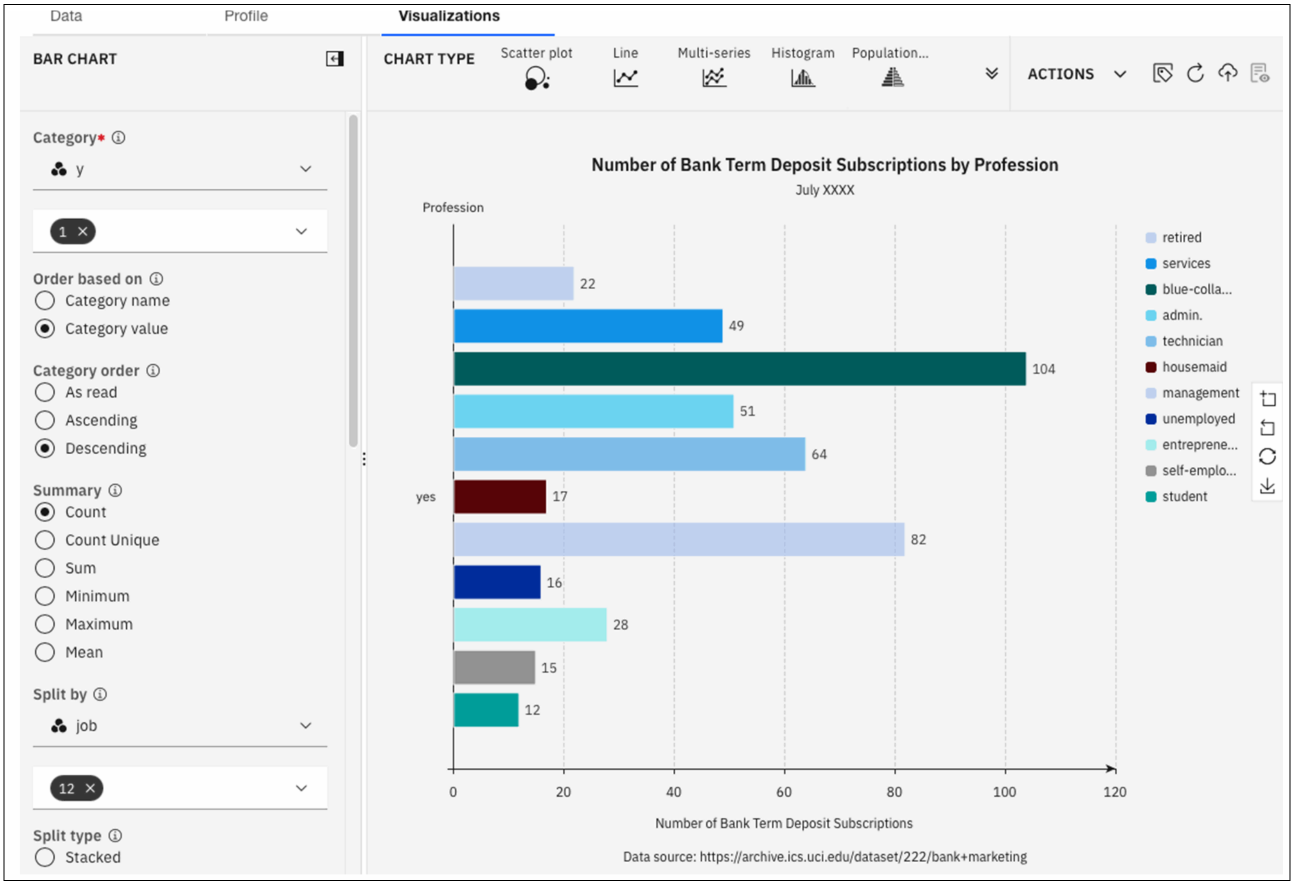
Basic setting

Theme name*: ABC_Banking_Institution_Theme

Color sequence

Figure 8-58 Applying the theme to a chart

Figure 8-59 shows the new theme that is applied to the bar chart. The data analyst can use the new theme in future data visualizations too.



BAR CHART

Category* ⓘ
y

1 ×

Order based on ⓘ
○ Category name
● Category value

Category order ⓘ
○ As read
○ Ascending
● Descending

Summary ⓘ
● Count
○ Count Unique
○ Sum
○ Minimum
○ Maximum
○ Mean

Split by ⓘ
job

12 ×

Split type ⓘ
○ Stacked

Figure 8-59 Data visualization with new theme applied

Exporting and sharing the data visualization

Now, the data analyst exports and shares the data visualization by completing the following steps:

1. The data analyst selects **Actions** → **Save Visualization to project**, as shown in Figure 8-60 on page 139. Now, users with access to the project can view and access this specific data visualization.

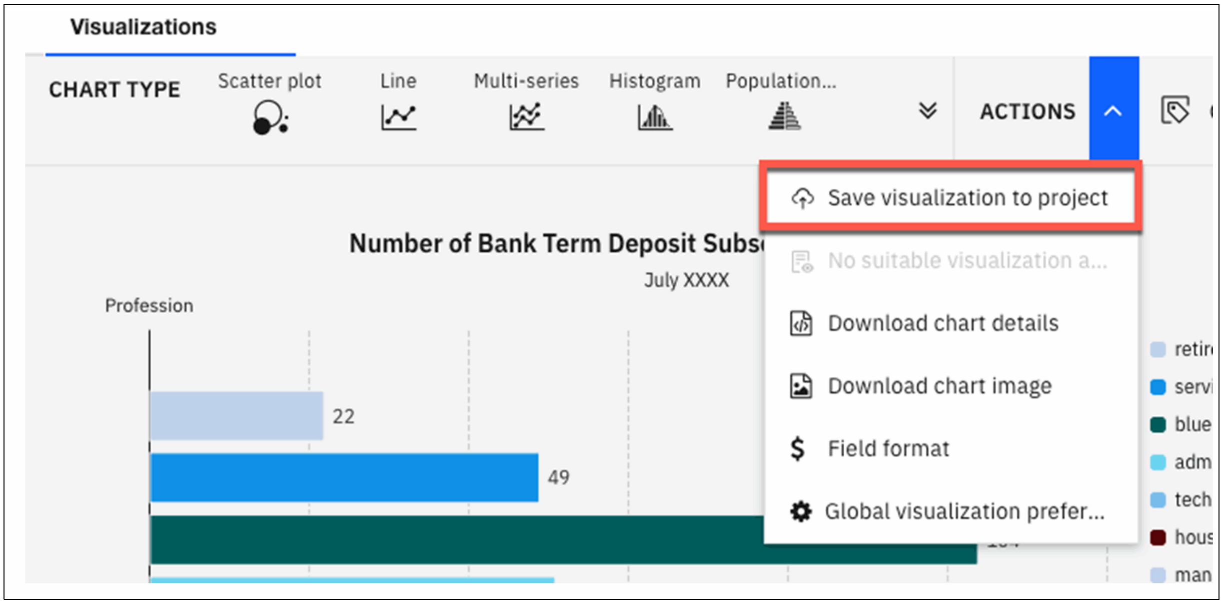
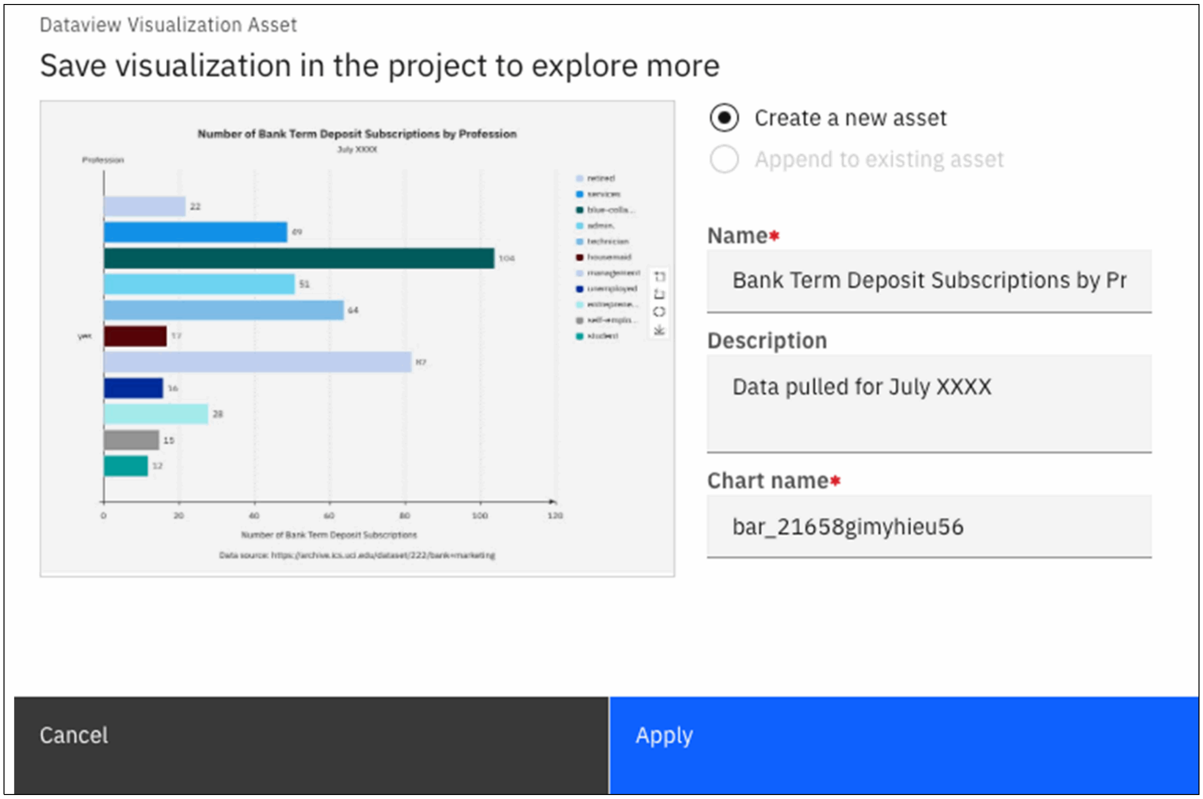


Figure 8-60 Save visualization to project

2. To save the data visualization to the project, the data analyst adds a name, description, and chart name, and then clicks **Apply**, as shown in Figure 8-61.


Dataview Visualization Asset

Save visualization in the project to explore more

Number of Bank Term Deposit Subscriptions by Profession
July XXXX
Profession
22
49
104
51
64
17
82
16
28
15
12
0
20
40
60
80
100
120
Number of Bank Term Deposit Subscriptions

● Create a new asset

○ Append to existing asset

Name*

Bank Term Deposit Subscriptions by Pr

Description

Data pulled for July XXXX

Chart name*

bar_21658gimyhieu56

Figure 8-61 Save visualization in project: Apply

3. The data analyst can also download the chart image locally by selecting **Actions** → **Download chart image**, as shown in Figure 8-62. This option is helpful when sharing with users who do not have access to the project, and for cases when the data visualization must be embedded into other mediums, such as a presentation.

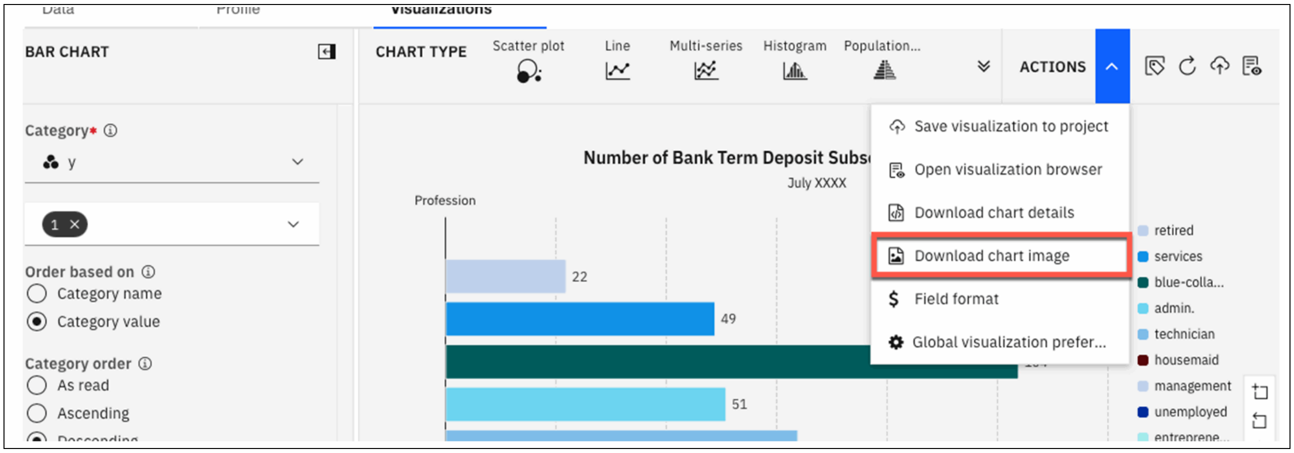
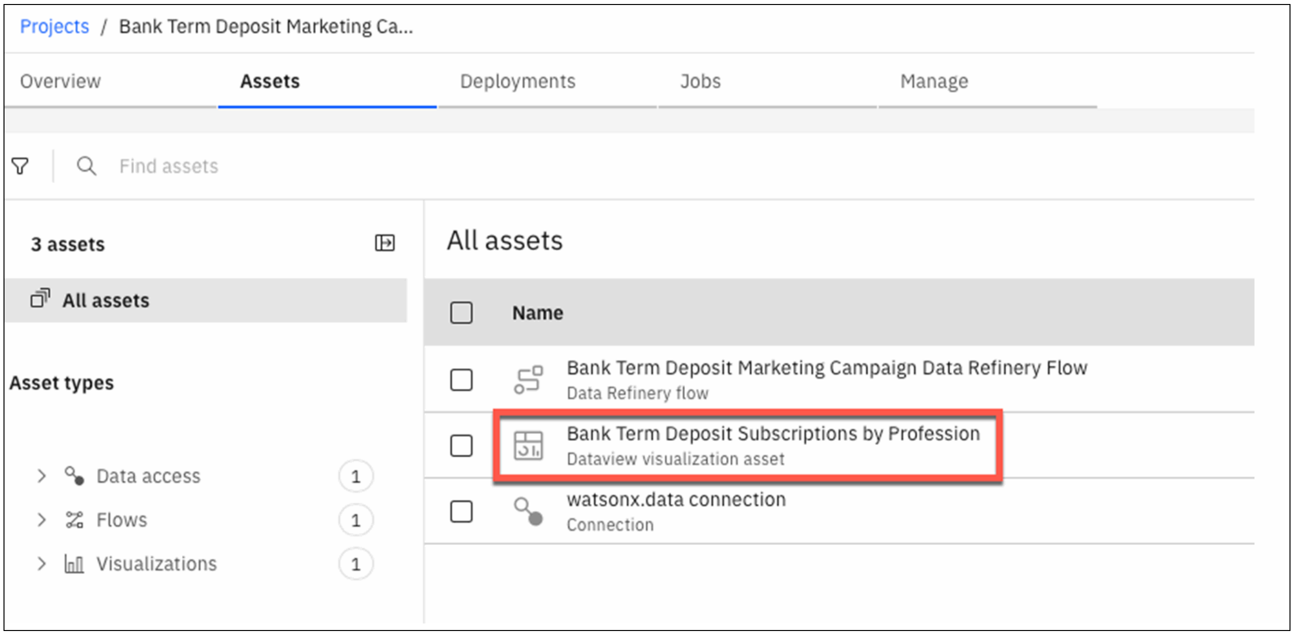


Figure 8-62 Download chart image menu

4. Figure 8-63 shows the data visualization asset on the **Assets** tab of the project.



☐	Name
☐	Bank Term Deposit Marketing Campaign Data Refinery Flow Data Refinery flow
☐	Bank Term Deposit Subscriptions by Profession Dataview visualization asset
☐	watsonx.data connection Connection

Figure 8-63 All assets: Bank Term Deposit Subscriptions by Profession

The data analyst selects this asset to view the data visualization from the perspective of the user to confirm that the data visualization appears as expected. Figure 8-64 on page 141 shows this view when the data visualization asset is opened.

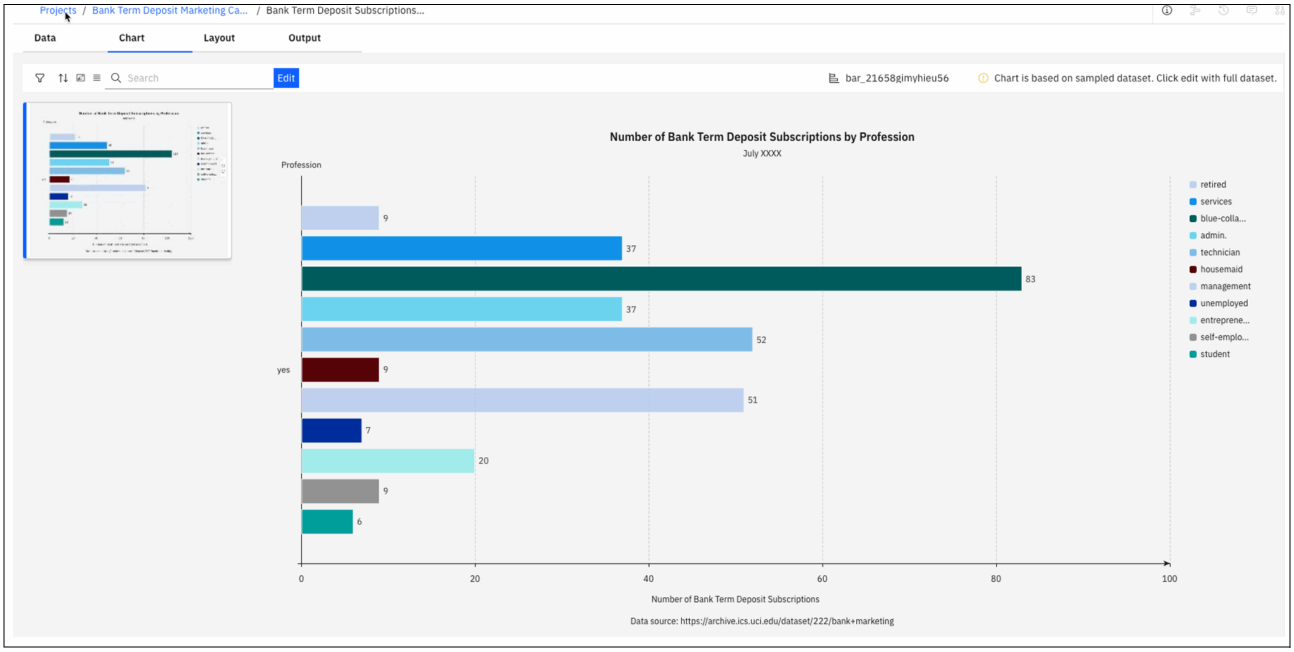


Figure 8-64 All assets: Bank Term Deposit Subscriptions by Profession - Chart

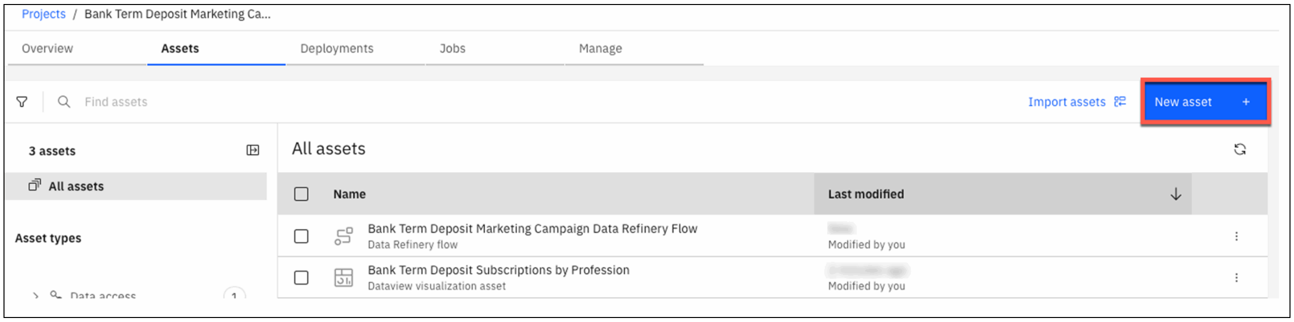
8.5 Building and developing machine learning models

The data connection in the IBM watsonx.ai project can also be used by data scientists. In this section, the data scientist on this initiative uses the IBM watsonx.data data connection in IBM watsonx.ai to build and develop prototype models to predict whether someone will purchase a bank term deposit.

8.5.1 Creating an AutoAI experiment

To build and evaluate multiple machine learning models, the data scientist chooses to use AutoAI within IBM watsonx.ai. AutoAI is a graphical tool that you can use to develop, display, and rank various model candidate pipelines by using a data set. To accomplish this task, complete the following steps:

1. To create an AutoAI experiment, the data scientist clicks **New asset** in the **Assets** tab of the Bank Term Deposit Marketing Campaign Analysis project in IBM watsonx.ai, as shown in Figure 8-65.

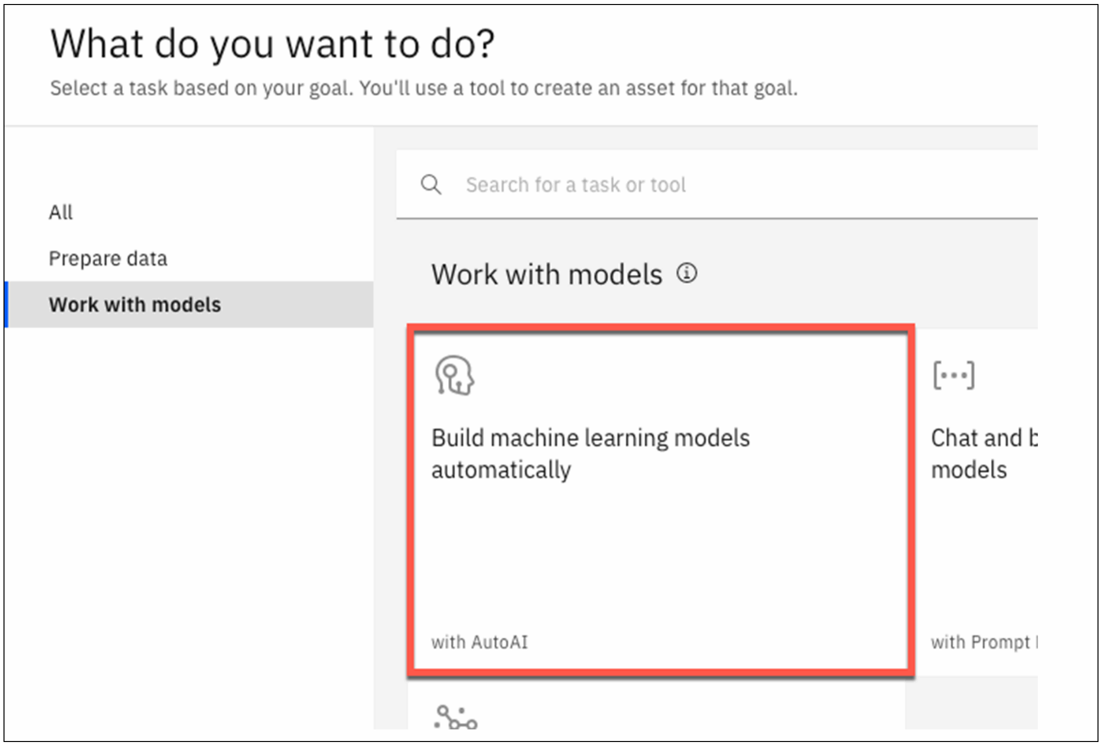


All assets

☐	Name	Last modified
☐	Bank Term Deposit Marketing Campaign Data Refinery Flow Data Refinery flow	[redacted] Modified by you
☐	Bank Term Deposit Subscriptions by Profession Dataview visualization asset	[redacted] Modified by you

Figure 8-65 New asset

2. The data scientist clicks **Work with models**, and then clicks **Build machine learning models automatically with AutoAI**, as shown in Figure 8-66.

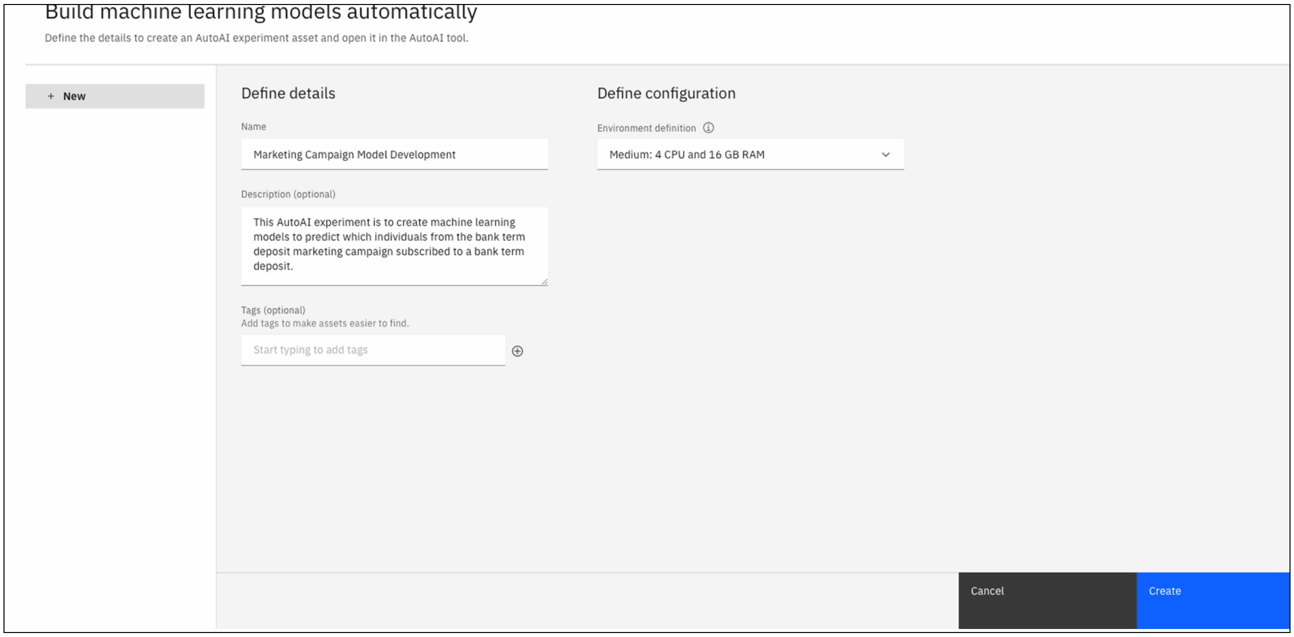


What do you want to do?

Select a task based on your goal. You'll use a tool to create an asset for that goal.

Figure 8-66 Build machine learning models automatically with AutoAI

3. The data scientist enters a name and description for the AutoAI experiment, as shown in Figure 8-67.



Build machine learning models automatically

Define the details to create an AutoAI experiment asset and open it in the AutoAI tool.

Define details

Name: Marketing Campaign Model Development

Description (optional): This AutoAI experiment is to create machine learning models to predict which individuals from the bank term deposit marketing campaign subscribed to a bank term deposit.

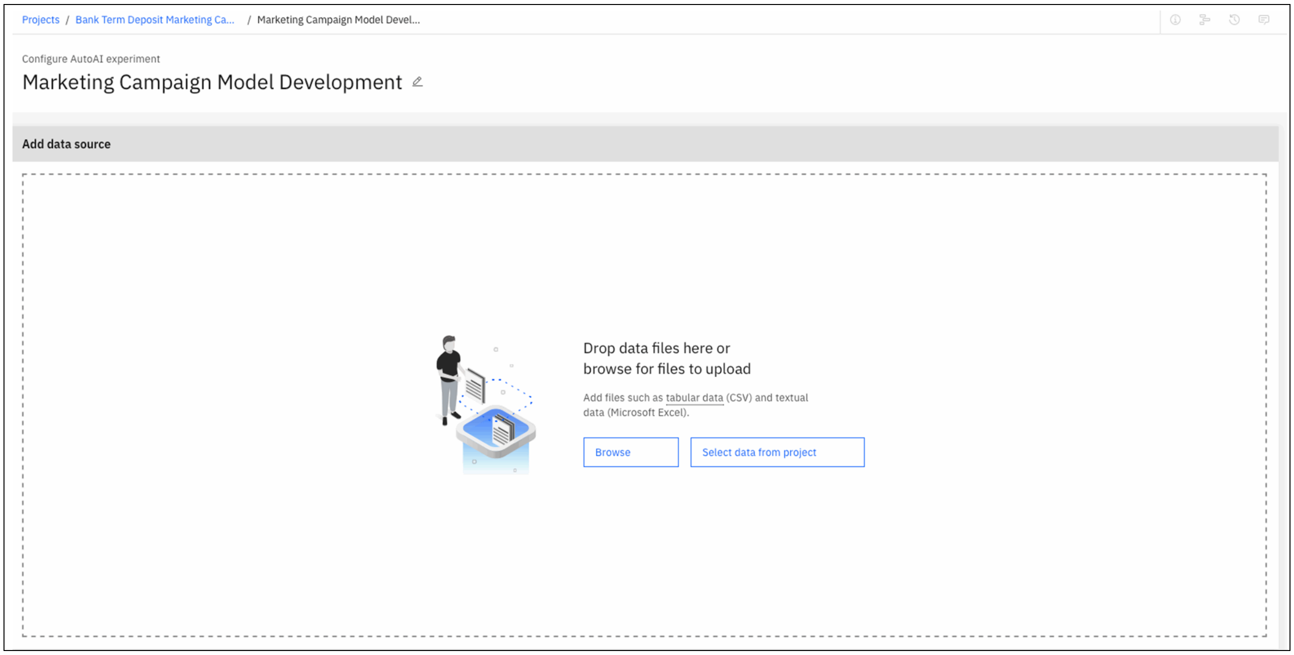
Tags (optional)
Add tags to make assets easier to find.

Define configuration

Environment definition: Medium: 4 CPU and 16 GB RAM

Figure 8-67 Build machine learning models automatically: Create

4. The data scientist adds data to the AutoAI experiment by clicking **Select data** from project, as shown in Figure 8-68.

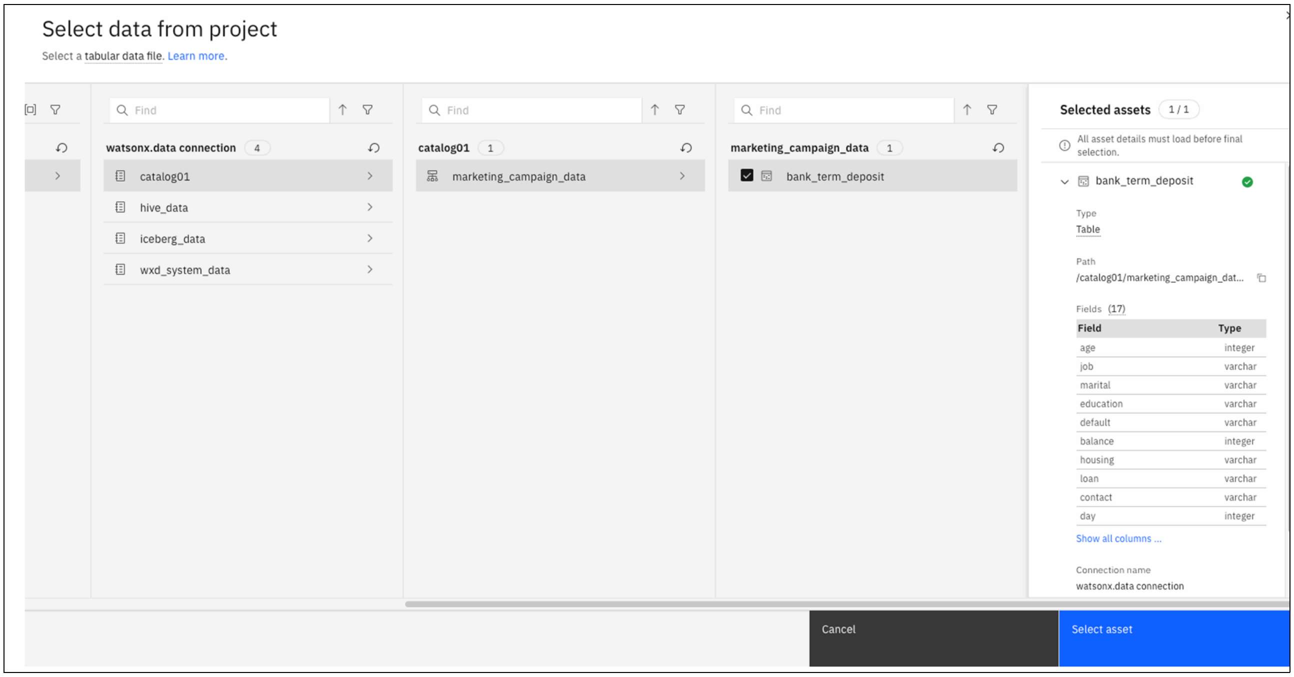

Projects / Bank Term Deposit Marketing Ca... / Marketing Campaign Model Devel...
Configure AutoAI experiment
Marketing Campaign Model Development
Add data source
Browse
Select data from project

Drop data files here or browse for files to upload

Add files such as tabular data (CSV) and textual data (Microsoft Excel).

Figure 8-68 Marketing Campaign Model Development: Selecting data from a project

5. The data scientist selects the bank_term_deposit table from the marketing_campaign_data schema, as shown in Figure 8-69.



All asset details must load before final selection.

bank_term_deposit

Type
Table

Path
/catalog01/marketing_campaign_dat...

Fields (17)

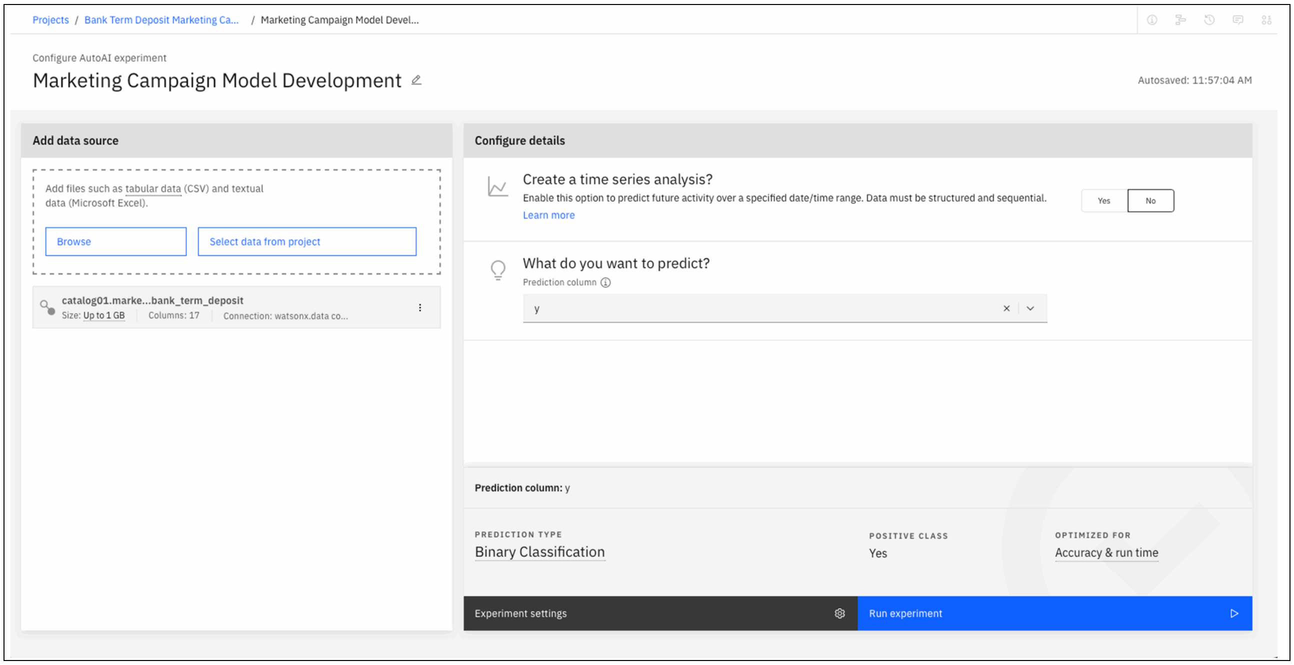
Field	Type
age	integer
job	varchar
marital	varchar
education	varchar
default	varchar
balance	integer
housing	varchar
loan	varchar
contact	varchar
day	integer

Show all columns ...

Connection name
watsonx.data connection

Figure 8-69 Selecting an asset

6. Once the data source is selected for the experiment, the data scientist configures the AutoAI experiment by setting “Create a time series analysis?” to **No**, and identifying **y** as the column to predict. They click **Experiment settings** to modify additional configuration settings for the experiment, as shown in Figure 8-70.



Projects / Bank Term Deposit Marketing Ca... / Marketing Campaign Model Devel...

Configure AutoAI experiment

Marketing Campaign Model Development

Autosaved: 11:57:04 AM

Add data source

Add files such as tabular data (CSV) and textual data (Microsoft Excel).

Browse | Select data from project

catalog01.marke...bank_term_deposit
Size: Up to 1 GB | Columns: 17 | Connection: watsonx.data co...

Configure details

Create a time series analysis?
Enable this option to predict future activity over a specified date/time range. Data must be structured and sequential.
Learn more

Yes [✓ No]

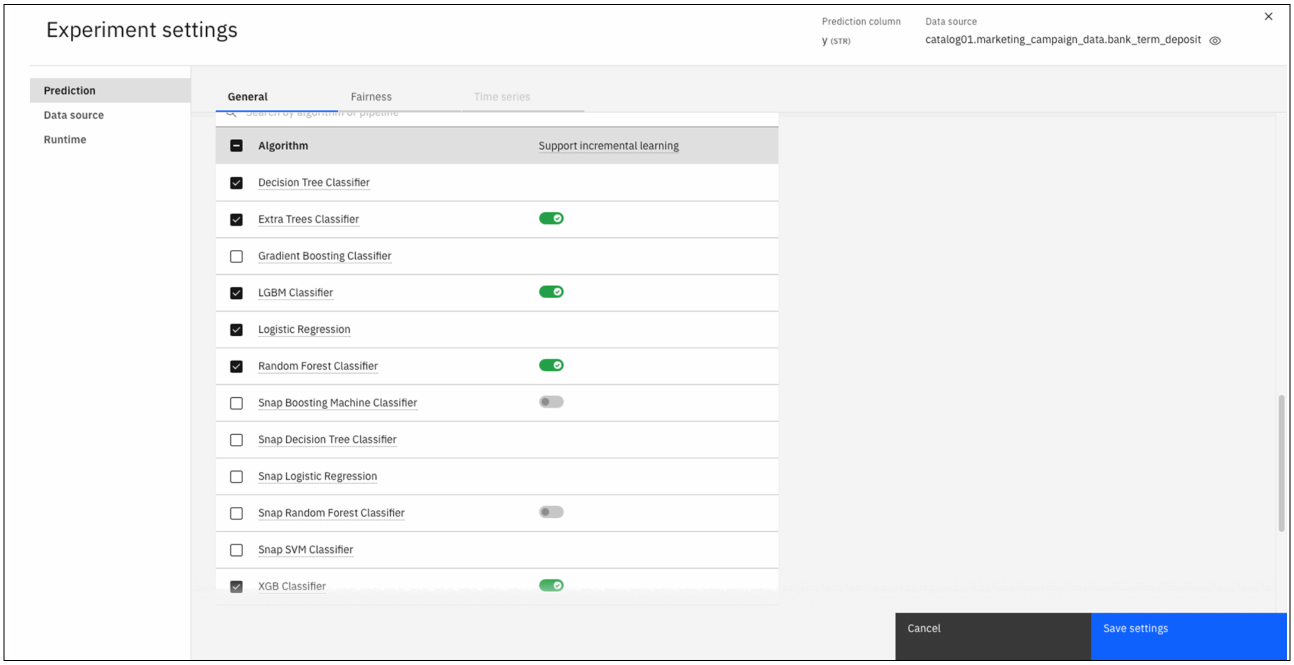
What do you want to predict?
Prediction column
y

Prediction column: y

PREDICTION TYPE	POSITIVE CLASS	OPTIMIZED FOR
Binary Classification	Yes	Accuracy & run time

Figure 8-70 Configure details: Experiment settings

7. The data scientist selects six algorithms for this experiment by selecting **Prediction** → **General** → **Algorithm**, as shown in Figure 8-71.



Experiment settings

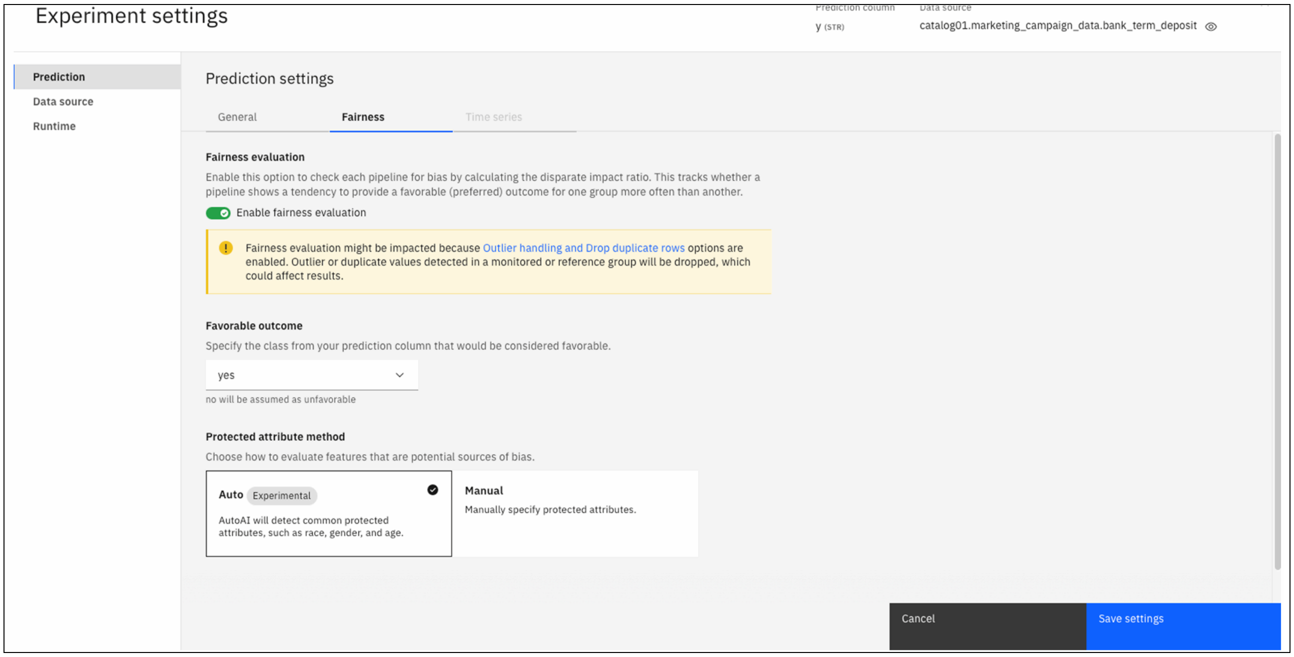
Prediction column: y (STR)

Data source: catalog01.marketing_campaign_data.bank_term_deposit

	Algorithm	Support incremental learning
☑	Decision Tree Classifier	
☑	Extra Trees Classifier	●
☐	Gradient Boosting Classifier	
☑	LGBM Classifier	●
☑	Logistic Regression	
☑	Random Forest Classifier	●
☐	Snap Boosting Machine Classifier	○
☐	Snap Decision Tree Classifier	
☐	Snap Logistic Regression	
☐	Snap Random Forest Classifier	○
☐	Snap SVM Classifier	
☑	XGB Classifier	●

Figure 8-71 Algorithm

8. Click the **Fairness** tab, and enable fairness evaluation. Enabling fairness evaluation helps ensures that results are not biased. The data scientist clicks **Save settings** to apply the changes, as shown in Figure 8-72.



Experiment settings

Prediction column y (STR) | Data source catalog01.marketing_campaign_data.bank_term_deposit

Prediction settings

Fairness evaluation

Enable this option to check each pipeline for bias by calculating the disparate impact ratio. This tracks whether a pipeline shows a tendency to provide a favorable (preferred) outcome for one group more often than another.

● Enable fairness evaluation

Fairness evaluation might be impacted because Outlier handling and Drop duplicate rows options are enabled. Outlier or duplicate values detected in a monitored or reference group will be dropped, which could affect results.

Favorable outcome

Specify the class from your prediction column that would be considered favorable.

yes

no will be assumed as unfavorable

Protected attribute method

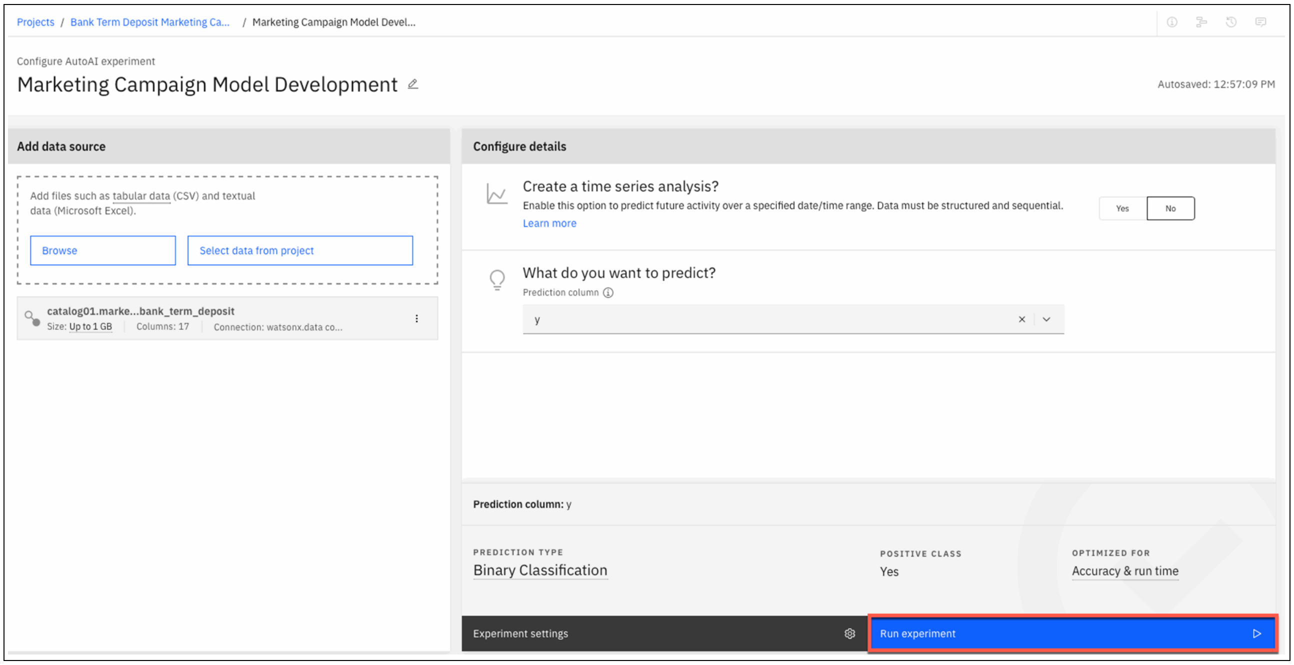
Choose how to evaluate features that are potential sources of bias.

Auto Experimental ● — AutoAI will detect common protected attributes, such as race, gender, and age.

Manual — Manually specify protected attributes.

Figure 8-72 Enable fairness evaluation: Save settings

9. The data scientist clicks **Run experiment** to begin the model development, as shown in Figure 8-73.



Projects / Bank Term Deposit Marketing Ca... / Marketing Campaign Model Devel...

Configure AutoAI experiment

Marketing Campaign Model Development

Autosaved: 12:57:09 PM

Add data source

Add files such as tabular data (CSV) and textual data (Microsoft Excel).

catalog01.marke...bank_term_deposit
Size: Up to 1 GB | Columns: 17 | Connection: watsonx.data co...

Configure details

Create a time series analysis?
Enable this option to predict future activity over a specified date/time range. Data must be structured and sequential.
Learn more

What do you want to predict?
Prediction column
y

Prediction column: y

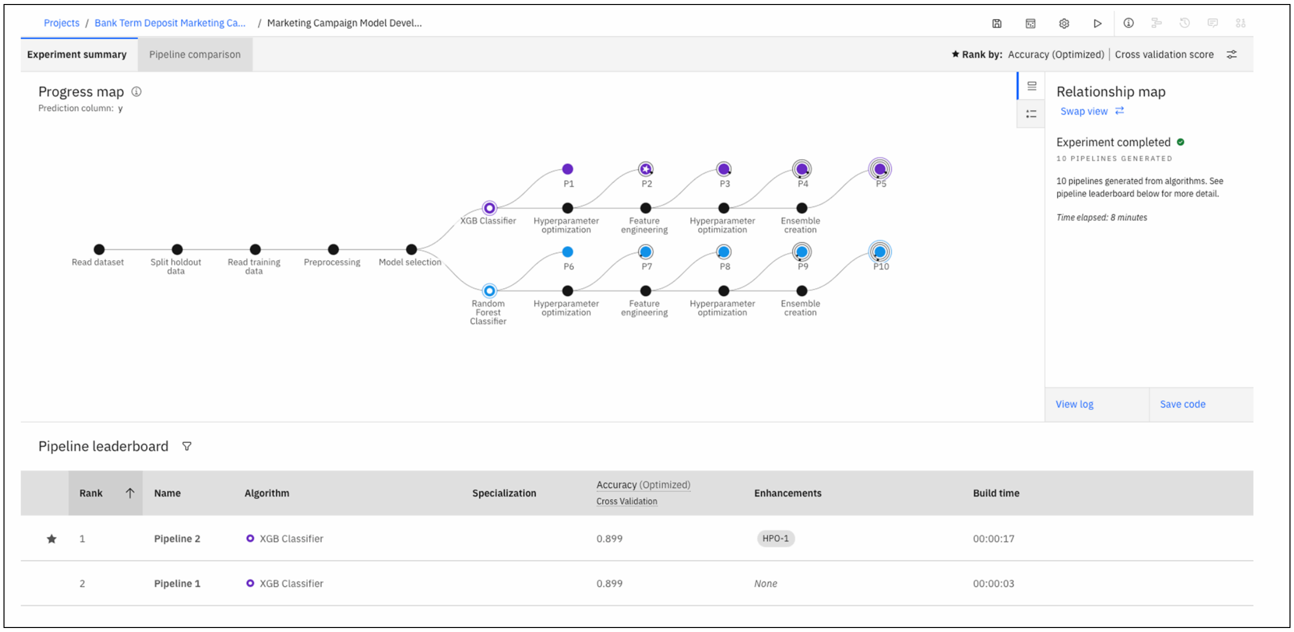
PREDICTION TYPE: Binary Classification

POSITIVE CLASS: Yes

OPTIMIZED FOR: Accuracy & run time

Figure 8-73 Run experiment

10. The AutoAI experiment takes a few minutes to complete. Once complete, the data scientist reviews the models generated by using the pipeline leaderboard, as shown in Figure 8-74 on page 146.



Projects / Bank Term Deposit Marketing Ca... / Marketing Campaign Model Devel...

Experiment summary | Pipeline comparison

★ Rank by: Accuracy (Optimized) | Cross validation score

Progress map ⓘ

Prediction column: y

Relationship map

Swap view ⇄

Experiment completed ✓

10 PIPELINES GENERATED

10 pipelines generated from algorithms. See pipeline leaderboard below for more detail.

Time elapsed: 8 minutes

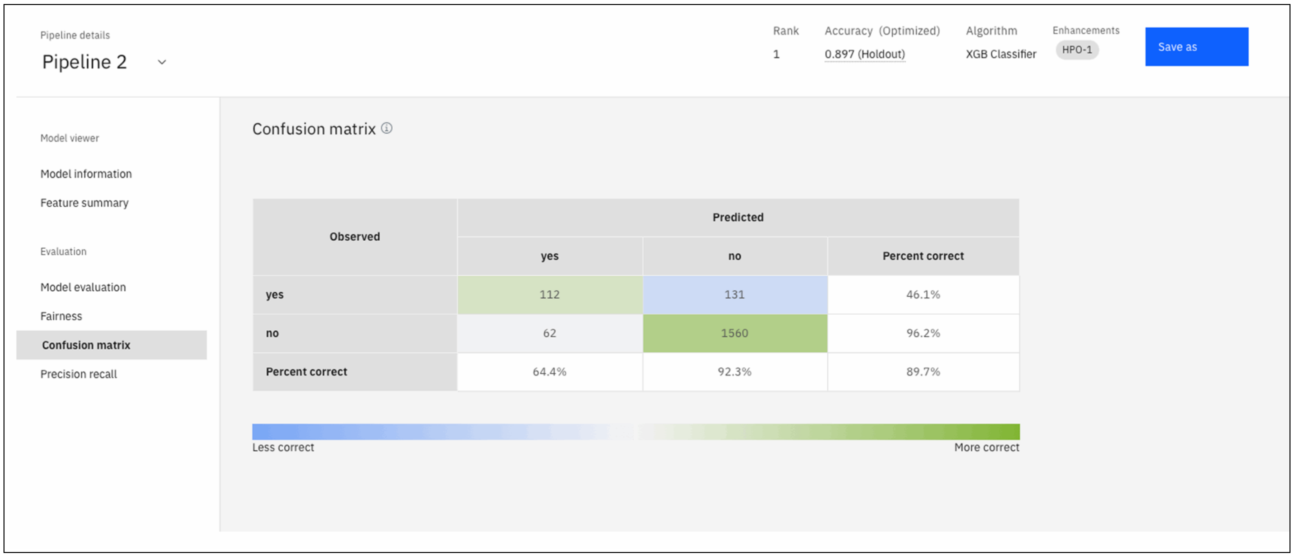
View log | Save code

Pipeline leaderboard

	Rank ↑	Name	Algorithm	Specialization	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 2	○ XGB Classifier		0.899	HPO-1	00:00:17
	2	Pipeline 1	○ XGB Classifier		0.899	*None*	00:00:03

Figure 8-74 Experiment summary: Pipeline leaderboard

11. The data scientist views additional model details of the highest ranked model, which is Pipeline 2, as shown in Figure 8-75.



Pipeline details

Pipeline 2

Rank	Accuracy (Optimized)	Algorithm	Enhancements
1	0.897 (Holdout)	XGB Classifier	HPO-1

Save as

Model viewer

- Model information
- Feature summary

Evaluation

- Model evaluation
- Fairness
- Confusion matrix
- Precision recall

Confusion matrix ⓘ

Observed	Predicted		
	yes	no	Percent correct
yes	112	131	46.1%
no	62	1560	96.2%
Percent correct	64.4%	92.3%	89.7%

Figure 8-75 Pipeline 2: Confusion matrix

12. There is an option to save the model as either a model or notebook, but the data scientist chooses to gather the results and share with the broader team before adjusting or deploying the model.

9

Adopting Milvus for RAG using IBM watsonx

By combining the generative power of large language models (LLMs) with the precision of information retrieval, Retrieval-Augmented (RAG) systems can deliver more accurate, relevant, and contextually rich outputs. This integration becomes even more powerful when leveraging robust vector databases like Milvus alongside comprehensive AI platforms such as IBM watsonx. This approach allows developers to build sophisticated RAG pipelines that efficiently manage and query massive datasets, enriching the capabilities of watsonx and unlocking new possibilities for intelligent applications.

This chapter discusses how to adopt Milvus for RAG using IBM watsonx and has the following sections:

9.1 Introduction

Retrieval-Augmented Generation (RAG) is a technique that enhances the performance of large language models (LLMs) by integrating them with a retrieval system. This approach helps improve the accuracy and relevance of the generated content by grounding it in external knowledge sources, such as documents, databases, or other types of structured and unstructured data.

RAG enables enterprises to transform resources such as policy manuals, training documents, or logs into a structured knowledge base that enhances LLMs. This approach benefits applications in customer support, on-site assistance, employee training, and developer productivity. By using RAG, companies reduce the risk of LLMs leaking sensitive data or producing inaccurate or misleading information. Additionally, RAG helps lower the computational and financial costs of running LLM-based systems in enterprise environments.

A key component in RAG systems is the use of a vector database, which stores data in the form of high-dimensional vectors. These databases are optimized for similarity search and allow LLMs to quickly find relevant information based on vector representations.

By combining vector databases with LLMs, RAG systems enable businesses to enhance the search and retrieval capabilities of their knowledge systems, driving more relevant and accurate results in various applications. This architecture, supported by platforms like IBM's watsonx, provides a powerful solution for managing large volumes of data and delivering high-quality AI outputs tailored to enterprise needs.

9.2 Key steps in the RAG workflow

The following are the key steps in the RAG workflow.

9.2.1 Step 1: Document ingestion

Document ingestion is the first step in the RAG workflow. It contains several aspects such as identifying sources, determining document types, setting ingestion frequency, configuring access permissions, gathering metadata, etc. Usually, the users may have company records, knowledge base or CRM systems in their internal databases and file systems. They may have existing tools to pull data from websites, blogs or online articles or through APIs in the existing services.

IBM watsonx.data provides a Spark engine which is a valuable tool in the document ingestion phase, especially for large volumes of documents. Spark supports a variety of data sources, including HDFS, S3, JDBC, and NoSQL databases, making it easy to collect documents from multiple origins. For instance, if you are collecting news articles from various online sources, you could setup a Spark job to:

- Use web scraping techniques to gather articles.
- Preprocess and clean the text using Spark DataFrame operations.
- Store the cleaned articles along with their metadata in the object store for further processing.

For more information, see Ingesting data by using Spark through the web console.

9.2.2 Step 2: Document chunking

Document chunking is essential in RAG. Large documents can contain information on multiple topics or subtopics. By breaking documents into smaller chunks, each representing a coherent piece of information, retrieval models can more accurately match relevant information to a query, reducing the chance of irrelevant sections appearing in search results.

Chunking can be done in various ways depending on the type of document, content structure, and retrieval system requirements. The fixed-length chunking usually splits the document into chunks based on a fixed number of tokens, words or characters. To mitigate information loss at chunk boundaries, you can employ a sliding window technique. This involves creating overlapping segments by moving a fixed-size window across the text. In certain scenarios, like legal documents or scientific articles, where context across sentences is crucial, the sliding window technique can be employed.

In other use cases such as well-structured documents like books or formal reports, you can leverage NLP sentence splitters or paragraph identifiers to chunk them based on semantics. IBM Natural Language Understanding services are very helpful for semantic chunking by providing text processing tools like sentence segmentation, paragraph splitting and entity recognition. For more information, see Natural Language Processing.

9.2.3 Step 3: Embedding generation

Embedding in RAG refers to the process of converting text (like a sentence, paragraph or document chunk) into a high-dimensional numerical vector that captures the semantic meaning of the text. These vectors, or “embeddings” are essential for efficient information retrieval and are a foundational component in RAG.

Embeddings transform textual data into a format that captures its meaning in a way that computers can understand and compare. Similar pieces of text (based on meaning, not just words) have embeddings that are closer together in vector space. For example, sentences like “*The cat is sleeping on the mat*” and “*A cat is napping on a rug*” would be represented by embeddings that are close to each other in that high-dimensional space.

When a user query is issued, it’s also transformed into an embedding. The retrieval system then finds the stored document embeddings that are closest to the query embedding in vector space, using similarity measures like cosine similarity, etc. Embeddings are typically generated by pre-trained models like BERT, or IBM embedding models in watsonx.ai. For more information, see Supported encoder foundation models in watsonx.ai.

9.2.4 Step 4: Vector storage and searching with Milvus

In RAG, vector storage and searching are crucial for handling embeddings efficiently. Milvus, an open-source vector database designed for AI applications, is highly suited for this purpose due to its performance, scalability, and efficient handling of high-dimensional vector data. Milvus is optimized for large-scale vector storage and retrieval, making it ideal for RAG scenarios where embeddings from extensive document corpora need to be managed and queried. Milvus supports distributed storage like AWS S3, IBM Cloud Object Storage, MinIO to store vector data, ensuring it can manage both large-scale storage and parallel processing. As data volume grows, additional compute nodes can be added to maintain performance. Embeddings used in RAG models are often high-dimensional, with hundreds to thousands of dimensions. Milvus supports these high-dimensional vectors, leveraging Approximate Nearest Neighbor (ANN) search algorithms to perform fast and accurate searches without the computational burden that exact searches on high-dimensional vectors would require.

In addition to vector similarity search, Milvus also supports hybrid search, combining vector similarity with traditional filtering based on scalar fields. These fields can include structured attributes such as tags, timestamps, and categories. This combination is particularly useful when both similarity to a query and specific contextual filters are required to narrow down results efficiently. For more information, see Working with Milvus.

9.3 Technical integration of IBM watsonx and Milvus

IBM watsonx.data has recently launched an integrated vector database, based on the open source Milvus, in the data lakehouse. Now, IBM watsonx customers can unify, curate and prepare vectorized embeddings for their generative artificial intelligence (gen AI) applications at scale across their trusted, governed data. The vector DB initially supported is based on open source vector DB Milvus. This supports up to 100 million vectors of 384 dimensions.

9.3.1 Architecture overview

IBM watsonx offers an interface for experimenting with various foundation models through engineered prompts. By incorporating watsonx.data Milvus as the vector database, you can enhance model accuracy and relevance by adding grounding documents as a knowledge base. These grounding documents, supported in formats like DOCX, PDF, PPTX, and TXT, are first converted into text embeddings by using pre-trained embedding models. Then, these embeddings are indexed by using Milvus vector database for efficient searching during prompt processing, ensuring more reliable and up-to-date responses.

Figure 9-1 on page 151 shows the architecture of watsonx RAG.

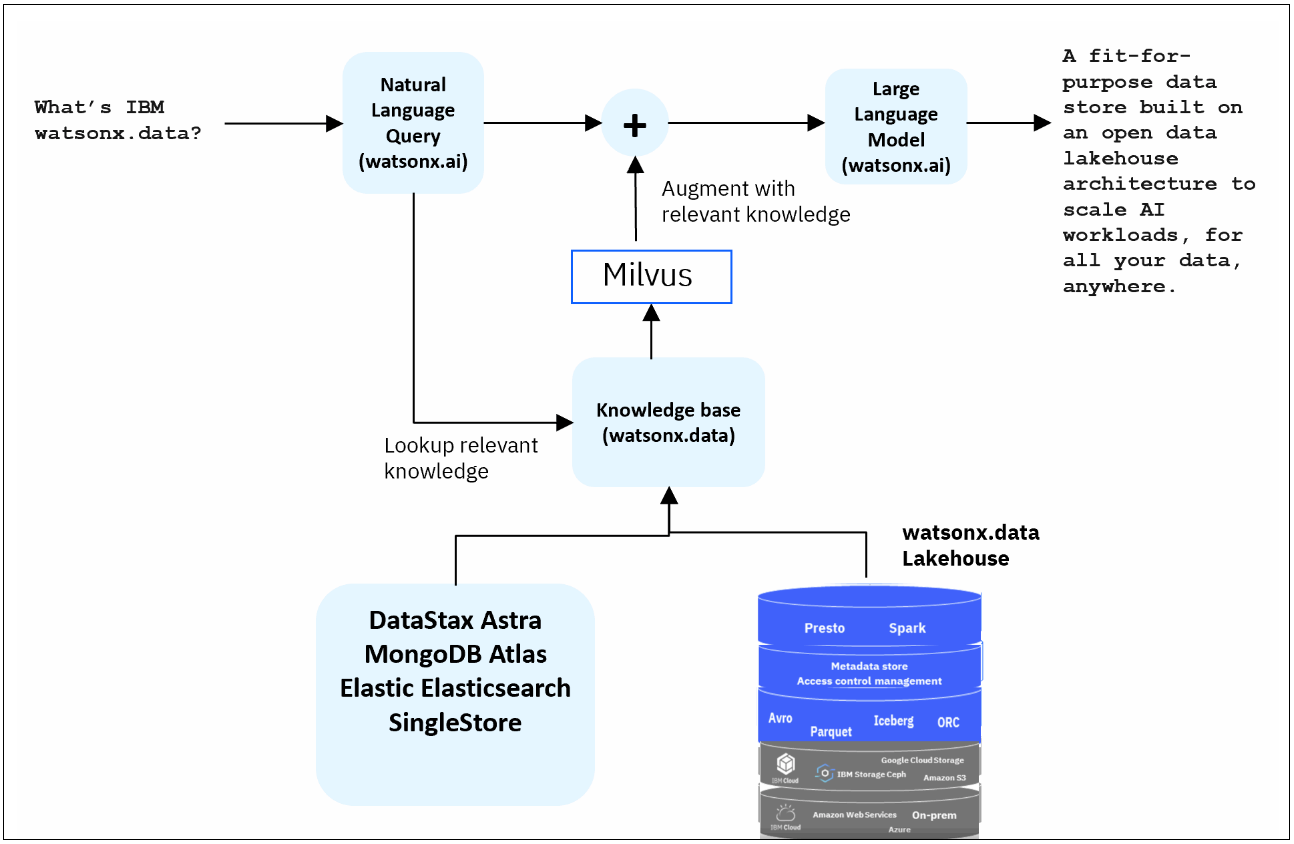

What's IBM watsonx.data?
Natural Language Query (watsonx.ai)
+
Large Language Model (watsonx.ai)
A fit-for-purpose data store built on an open data lakehouse architecture to scale AI workloads, for all your data, anywhere.
Augment with relevant knowledge
Milvus
Knowledge base (watsonx.data)
Lookup relevant knowledge
watsonx.data Lakehouse
DataStax Astra
MongoDB Atlas
Elastic Elasticsearch
SingleStore
Presto
Spark
Metadata store
Access control management
Avro
Parquet
Iceberg
ORC
IBM Cloud
IBM Storage Ceph
Google Cloud Storage
Amazon S3
IBM Cloud
Amazon Web Services
On-prem
Azure

Figure 9-1 Architecture of watsonx RAG

9.3.2 Code examples

Example 9-1 demonstrates how to ingest documents using Spark from the IBM documentation website, followed by converting the PDF files to text for embeddings.

Example 9-1 Code example - 1

```
from pyspark.sql import SparkSession
import os
spark = SparkSession.builder \
    .appName("Download watsonx.data PDF documentation") \
    .getOrCreate()
def download_pdf(url, local_path):
    response = requests.get(url)
    if response.status_code == 200:
        with open(local_path, 'wb') as file:
            file.write(response.content)
        print(f"PDF downloaded successfully to {local_path}")
    else:
        print(f"Failed to download PDF. Status code: {response.status_code}")
pdf_url =
"https://www.ibm.com/support/pages/system/files/inline-files/IBM%20watsonx.data%20
version%202.0.3.pdf"
local_file_path = "wxd_doc_pdf.pdf"
download_pdf(pdf_url, local_file_path)
spark.stop()
```

```
asset_li=wslib.assets.list_assets("data_asset")
wslib.download_file("wxd_doc_pdf")
doc = fitz.open("wxd_doc_pdf")
pdf_text = ""
for page in doc:
    pdf_text += page.get_text()
```

Example 9-2 demonstrates how to chunk a document by paragraph.

Example 9-2 Code example - 2

```
def chunk_by_paragraphs(text):
    paragraphs = text.split("\n\n") # Assuming paragraphs are separated by two
newlines
    return [p.strip() for p in paragraphs if p.strip()]
chunks = chunk_by_paragraphs(pdf_text)
```

Example 9-3 demonstrates how to generate embeddings using IBM watsonx.data Milvus connection.

Example 9-3 Code example - 3

```
connections.connect(alias="default",
                    host=url,
                    port=port,
                    user=apiuser,
                    password=apikey,
                    secure=True)
collection_description = 'wxd docs pdf'
collection_name = 'wxd_documentation'
# Create collection - define fields + schema
fields = [
    FieldSchema(name="document_id", dtype=DataType.INT64), # Document Id
    FieldSchema(name="chunk_id",  dtype=DataType.INT64), # Chunk Id
    FieldSchema(name="embedding", dtype=DataType.FLOAT_VECTOR, dim=384),
]
# Create a schema
schema = CollectionSchema(fields, collection_description)
# Create a collection
collection = Collection(collection_name, schema)
# Create index
index_params = {
        'metric_type':'L2',
        'index_type':"IVF_FLAT",
        'params':{"nlist":2048}
}
collection.create_index(field_name="vector", index_params=index_params)
for i in range(len(article_titles)):
    # Create vector embeddings + data
    model = SentenceTransformer('sentence-transformers/all-minilm-l12-v2') # 384
dim
    passage_embeddings = model.encode(article_chunks[i])
    basic_collection = Collection(collection_name)
    data = [
        article_chunks[i],
        article_titles[i],
        passage_embeddings
    ]

    out = basic_collection.insert(data)
    basic_collection.flush()  # Ensures data persistence
```

Example 9-4 demonstrates how to perform a similarity search with Milvus.

Example 9-4 Code example - 4

```
def query_milvus(query, num_results):

    # Vectorize query
    model = SentenceTransformer('sentence-transformers/all-minilm-l12-v2') # 384
dim
    query_embeddings = model.encode([query])
    # Search
    search_params = {
        "metric_type": "L2",
        "params": {"nprobe": 5}
    }
    results = basic_collection.search(
        data=query_embeddings,
        anns_field="vector",
        param=search_params,
        limit=num_results,
        expr=None,
        output_fields=['article_text'],
    )
    return results
```

For more information, see Getting started with watsonx.data Spark use cases.

9.3.3 Benefits of using IBM watsonx for RAG

IBM watsonx for RAG has the following benefits:

Scalability

IBM watsonx.data adopts the architecture of separation of computation and storage. It can handle large-scale datasets with Milvus's distributed vector database for fast querying. The Milvus service can be added in watsonx.data with multiple sizings based on user's need. The Starter size can support 1 million vectors and the Large size can support 100 million vectors. If a user requires more vectors, they can work with IBM to customize their watsonx.data instance to support them.

Security

IBM watsonx.data provides enterprise-grade fine-grained access control to protect the data. Chapter 4 describes the general guidance on access controls. Milvus is one of the services in watsonx.data and inherit the security consideration in the platform level as well as the fine grained access control particularly on Milvus data objects such as databases, collections, partitions.

High-quality embeddings

IBM watsonx.ai provides quite a few embedding models including IBM own modes such as IBM Slate-30m embedding model and IBM Slate-125m embedding model, and some embedding models from third-party such as all-MiniLM-L6-v2 from open source NLP and computer vision community provided by Hugging Face.

Enhanced retrieval efficiency

IBM watsonx.data Milvus provides various search algorithms such as Sparse vector search, Multi-vector and Hybrid search, and Grouping search to improve the retrieval efficiency as well as the accuracy.

9.4 Summary

In this use case, we introduce RAG and outline the key steps involved in the RAG workflow, including document ingestion, chunking, embedding generation, vector storage and searching. We also demonstrate how to implement RAG using IBM watsonx, with a practical example. IBM watsonx is an integrated platform for AI and data applications that ensures high scalability and performance, enhanced security and governance, and continuous development with cutting-edge technology.

10

Data and AI modernization strategy in banking use case

By leveraging data-driven insights, banks can enhance customer experience, optimize operations, and gain a competitive edge. This chapter discusses an architecture for modernized data management pattern in banking.

This chapter has the following sections:

10.1 Introduction

The recent surge in generative AI, powered by large language models and agentic frameworks, has significantly increased the demand for training on massive datasets. Enterprises across industries are modernizing client interactions with human-like AI interfaces to gain a competitive edge. Effective data management is crucial to fuel this transformation.

Enterprise banks aim to revolutionize online and mobile banking through advanced analytics and real-time insights. Key strategies include:

- **Personalized banking:** Tailoring services like micropayments, cross-selling, and upselling to individual customer needs.
- **Instant transactions:** Enabling swift and efficient fund transfers.
- **Risk mitigation:** Implementing robust early warning systems and fraud detection measures.

By leveraging data-driven insights, banks can enhance customer experience, optimize operations, and gain a competitive edge.

To effectively execute real-time use cases, data attributes such as authenticity, lineage, governance, PII protection, aggregation, and open formats are essential. By ensuring data quality and security, banks can build trust with both existing and potential customers, ultimately enhancing their service offerings.

10.2 Data lakehouses: Empowering data-driven decisions in banking

Banks have traditionally relied on data warehouses for business intelligence workloads. While fraud detection on edge devices often involves real-time streaming data, large-scale data analysis (petabytes and exabytes) typically relies on data warehouses and data lakes, which can present challenges in terms of cost and performance.

The banking sector, traditionally cautious about cloud-based solutions, is increasingly embracing data lakehouses. These platforms, built on open data formats and lineage standards, offer improved control and flexibility in data management:

- Data warehouse(s) - Mature, mostly proprietary solutions optimized for high-performance aggregation and stringent service-level agreements (SLAs).
- Data lake(s) - Distributed storage systems, often Hadoop-based, suitable for large-scale batch workloads but with limitations.
- Data lakehouse(s) - A convergence of data warehouses and data lakes, leveraging open data processing engines and table formats for efficient analytical and batch workloads.
- Data lineage - A combination of open and proprietary standards to track data origins and transformations.
- Data governance - AI-powered semantic extraction and search to enhance data understanding and control.

10.3 Data modernization pattern in banking

Decades of heavy investment in optimization have led to stringent SLAs imposed by public sector governing bodies on banking use cases. While traditional warehouses effectively met these SLAs in the past, the increasing scale of data in certain use cases has challenged their cost-effectiveness and ability to maintain SLAs for downtime, recovery, and response times.

10.3.1 Current pattern in banking on analytic use cases

Data management typically involves three key layers:

- **Data ingestion services:** Both streaming and batch processing are used to ingest data.
- **Data stores:** Data is stored in various formats, including data lakes and data warehouses.
- **Lineage and governance:** While less emphasized due to a lack of well-integrated services, lineage and governance are crucial for data quality and compliance.

Figure 10-1 shows current pattern in banking on analytic use cases.

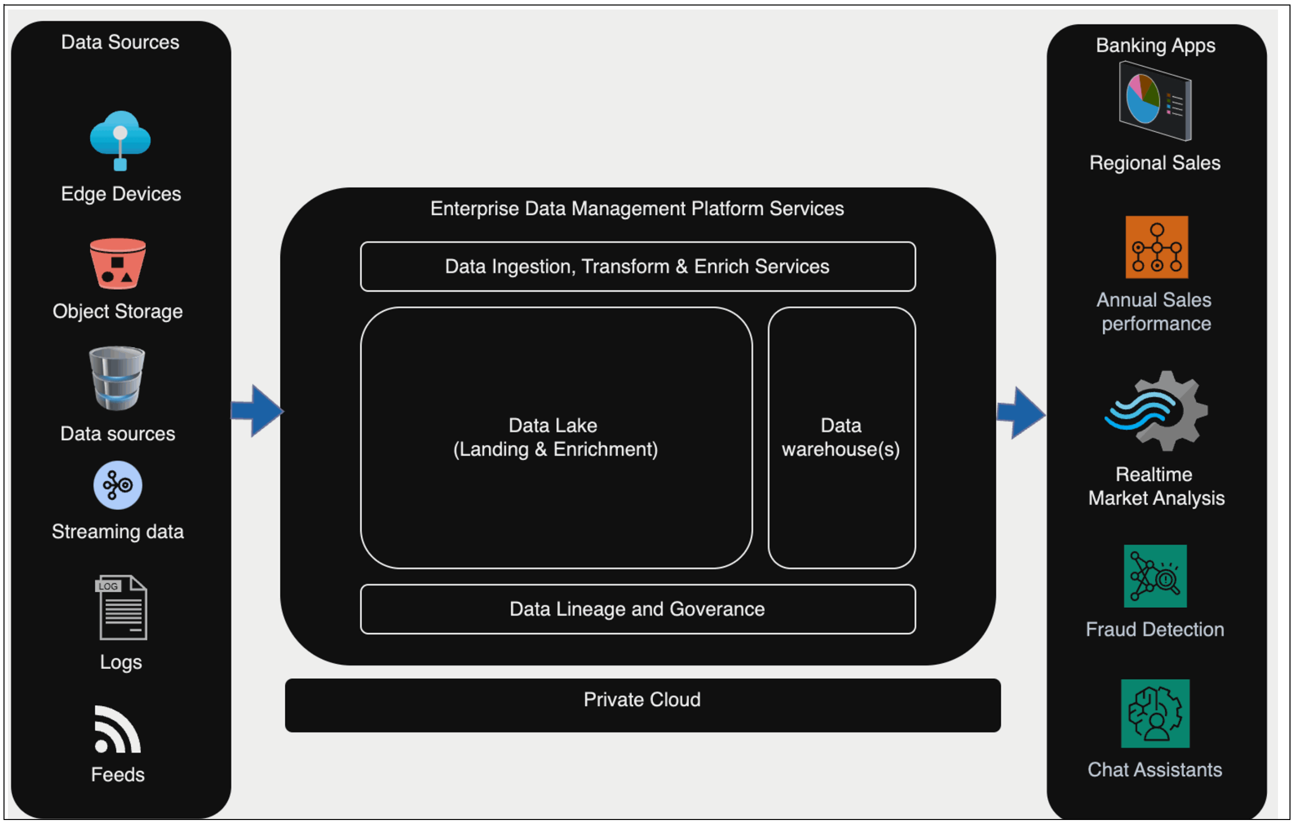


Figure 10-1 Current pattern in banking on analytic use cases

10.4 Modernized data management pattern in banking on analytic use cases

While the architecture flow has standardized over the years, some of the key differentiates are:

- Data lakehouse for scale at its core.
- Preserving existing investments to iteratively evolve then lift and shift into data lakehouse based architecture.
- Platform approach of well-integrated services across data and AI with governance and lineage across.
- AI powered semantic enrichment.
- Intelligent and simple AI assisted embedded services.

Figure 10-2 shows modernized data management pattern in banking on analytic use cases.

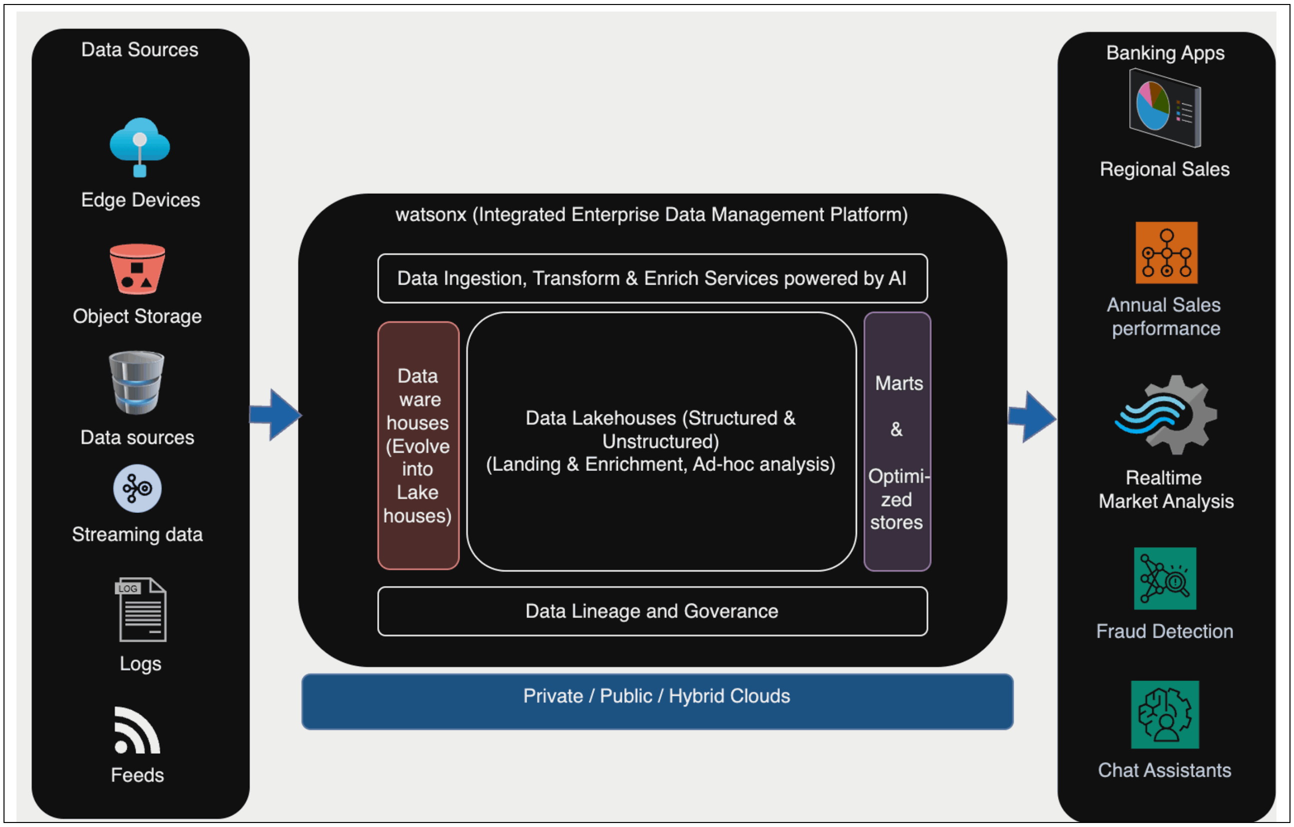

Data Sources
Edge Devices
Object Storage
Data sources
Streaming data
LOG
Logs
Feeds
watsonx (Integrated Enterprise Data Management Platform)
Data Ingestion, Transform & Enrich Services powered by AI
Data ware houses (Evolve into Lake houses)
Data Lakehouses (Structured & Unstructured) (Landing & Enrichment, Ad-hoc analysis)
Marts & Optimi-zed stores
Data Lineage and Goverance
Private / Public / Hybrid Clouds
Banking Apps
Regional Sales
Annual Sales performance
Realtime Market Analysis
Fraud Detection
Chat Assistants

Figure 10-2 Modernized data management pattern in banking on analytic use cases

List of banking use cases powered by the pattern shown in Figure 10-2 are as follows:

- Use case 1: Real-time business analytics
- Use case 2: Early warning system and fraud detection
- Use case 3: Anti-money laundering

10.5 Conclusion

When modernizing banking applications, it is crucial to safeguard existing investments while evolving beyond a lift-and-shift approach. Leverage watsonx, powered by open engines and data formats, AI-driven data services for enrichment and governance across the entire stack.

Related publications

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

IBM Redbooks

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

- *Simplify Your AI Journey: Ensuring Trustworthy AI with IBM watsonx.governance*, SG24-8573
- *Simplify Your AI Journey: Unleashing the Power of AI with IBM watsonx.ai,* SG24-8574
- *Unlocking Data Insights and AI: IBM Storage Ceph as a Data Lakehouse Platform for IBM watsonx.data and Beyond*, SG24-8563

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:

- watsonx.data IBM Documentation

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse

SG24-8570-00

ISBN 0738461954

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with

SG24-8570-00

ISBN 0738461954

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM

SG24-8570-00

ISBN 0738461954

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM

(0.2”spine)
0.17”<->0.473”
90<->249 pages

(0.1”spine)
0.1”<->0.169”
53<->89 pages

Redbooks

Simplify Your AI Journey: Hybrid, Open Data

SG24-8570-00

ISBN 0738461954

IBM

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

Redbooks

Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM watsonx.data

SG24-8570-00

ISBN 0738461954

IBM

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

SG24-8570-00

ISBN 0738461954

Printed in U.S.A.