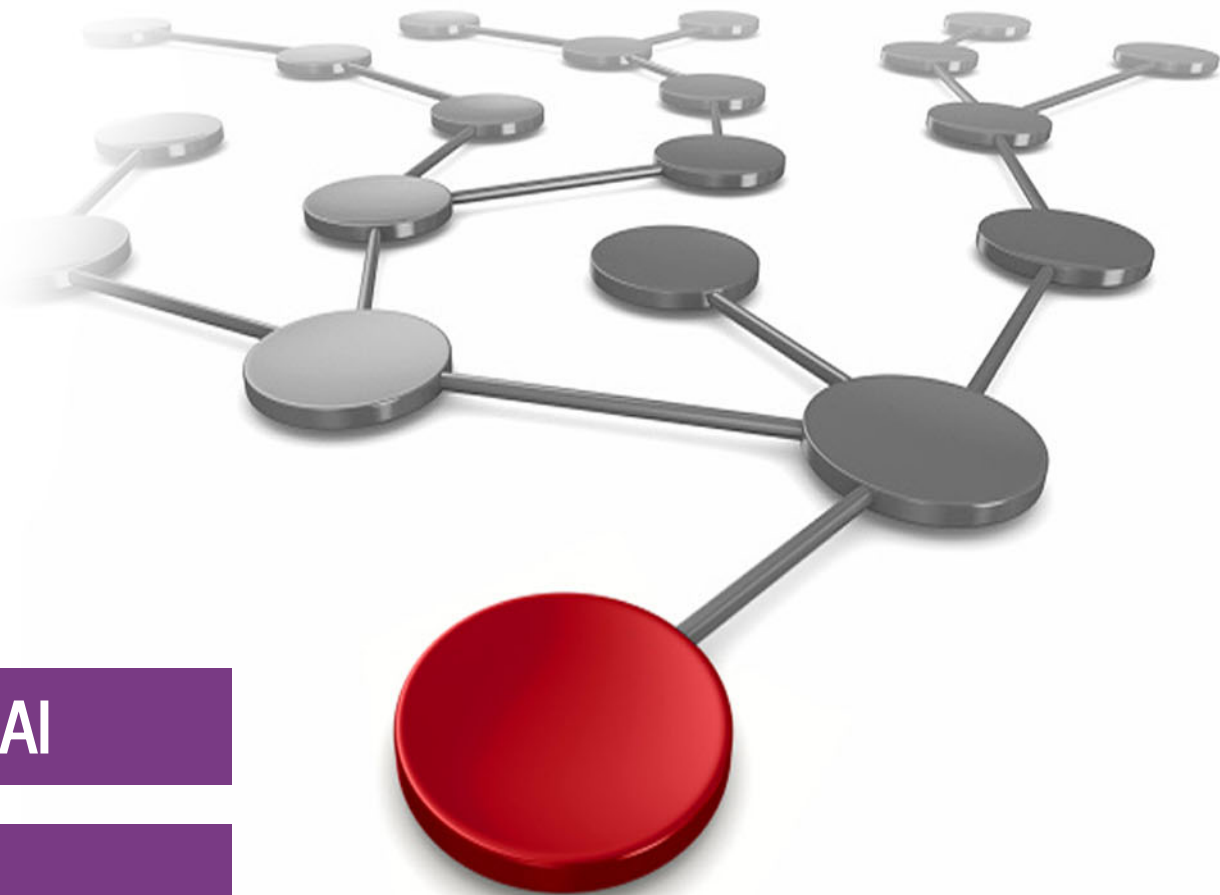


Next-generation Enterprise AI Solutions with IBM watsonx.ai and IBM Storage Scale

Qais Noorshams
Chinmaya Mishra
Harald Seipp
Sailendu Patra
Dietmar Fischer
Kedar Karmarkar
Mathias Defiebre



Data and AI

Cloud



IBM Redbooks

**Next-generation Enterprise AI Solutions with IBM
watsonx.ai and IBM Storage Scale**

March 2026

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

First Edition (March 2026)

This edition applies to IBM Storage Scale System Version 6, Release 0, Modification 0.

© Copyright International Business Machines Corporation 2026. All rights reserved.

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

Contents

Notices	v
Trademarks	vi
Preface	vii
Authors	vii
Now you can become a published author, too!	viii
Comments welcome	ix
Stay connected to IBM Redbooks	ix
Chapter 1. Introduction	1
1.1 The evolution of storage in the age of artificial intelligence	2
1.2 Benefits of IBM Storage Scale Global Data Platform	4
1.3 IBM watsonx ecosystem overview	5
Chapter 2. Solution architecture	7
2.1 Use cases for IBM Storage Scale with watsonx.ai	8
2.2 Solution architecture	8
2.3 Architecture variations	10
2.4 Benefits of IBM Spectrum Scale for AI use cases	12
2.5 IBM Storage Scale and IBM watsonx.data	13
Chapter 3. Planning and sizing	15
3.1 Hardware requirements	16
3.1.1 Compute cluster	16
3.1.2 Storage cluster	16
3.2 Network requirements	17
3.3 Software requirements	17
3.4 Sizing guidelines	18
3.4.1 watsonx.ai	18
3.4.2 IBM watsonx.data	19
3.4.3 IBM Storage Scale System	19
Chapter 4. Configuring the solution	21
4.1 Configuring IBM Storage Scale System and IBM Storage Scale	22
4.2 Configuring IBM Storage Scale CNSA	24
4.3 IBM Storage Scale CES S3	25
4.3.1 Installing and configuring the IBM Storage Scale S3 service	25
4.3.2 DNS load balancer for S3	26
4.3.3 Configuring file sets as backing storage for S3 buckets	27
4.3.4 Creating S3 buckets	27
4.4 Configuring data abstraction and acceleration	29
4.5 Defining IBM Storage Scale S3 buckets to IBM watsonx.data	32
4.6 Adding a Milvus vector database service to IBM watsonx.data	34
Chapter 5. Examples, use cases, and solution application	37
5.1 Working with IBM watsonx.ai	38
5.1.1 GUI for an interactive workflow	38
5.1.2 Notebook for a programmatic workflow	40
5.2 Use case: Robust entity extraction and summarization with Docling	40

5.2.1 High-level workflow overview	41
5.2.2 Financial summary example	41
5.3 Use case: Bulk data ingest and query evaluation with Open RAG Benchmark	45
5.3.1 Data ingest	46
5.3.2 RAG query	48
5.4 Use case: RAG with a role-based content filter on IBM Storage Scale.	51
5.4.1 Data formats and data preparation	51
5.4.2 Model setup	53
5.4.3 RAG queries	54
Abbreviations and acronyms	59
Related publications	61
IBM Redbooks	61
Online resources	61
Help from IBM	62

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®	IBM Watson®	watsonx®
Granite®	IBM watsonx®	watsonx Code Assistant®
IBM®	Orchestrate®	watsonx Orchestrate®
IBM Cloud®	Redbooks®	watsonx.ai®
IBM Spectrum®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

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

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

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

Preface

This IBM Redpaper describes the IBM® solution for using IBM Storage Scale as enterprise storage with IBM watsonx.ai®. The paper explains how IBM watsonx.ai applications can use the enterprise-storage features and functions that are provided by IBM Storage Scale.

This Redpaper is intended for technical professionals, including customers, consultants, technical support staff, IT architects, and IT specialists who are responsible for delivering artificial intelligence (AI) solutions. This Redpaper is relevant for the following audiences:

- ▶ Technical professionals who design and implement watsonx.ai solutions.
- ▶ IBM Storage Scale customers who plan to implement watsonx.ai solutions.

Authors

This paper was produced by a team working at IBM Redbooks, Tucson Center.

Qais Noorshams is a senior software engineer and solution architect within the IBM Storage Scale organization. Since joining IBM in 2015, he has held technical and leadership roles in international software development projects. He is a certified Expert Developer, IBM Recognized Speaker, and IBM Recognized Teacher. His work includes more than 20 granted patents, more than 15 peer-reviewed publications, and multiple IBM published newsletters and articles. He holds a diploma degree (Dipl.-Inform.) and a PhD degree (Dr.-Ing.) in computer science from the Karlsruhe Institute of Technology in Germany.

Chinmaya Mishra is a software architect in the Big Data & Analytics team within the IBM Storage Scale organization in IBM India Systems Development Labs. He joined IBM India in 2001 and has held various technical and leadership roles in software product and solutions development teams across IBM India and IBM US. His areas of expertise include transaction processing, operating systems, cloud-native solutions as a service, high-performance clustered file systems, and data and analytics solutions. He holds a Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Kharagpur.

Harald Seipp is a Senior Technical Staff Member and principal solution architect with IBM Client Engineering EMEA. He is responsible for designing storage cloud and SDS architectures and developing solutions that meet complex client requirements. He works in the areas of Kubernetes, Red Hat OpenShift, and object-storage solutions and implements first-of-a-kind systems with clients during proof-of-experience pilot engagements.

Sailendu Patra is a solution architect for Data and AI in IBM Client Engineering who is based in Bangalore, India. He holds a Bachelor of Technology degree from BPUT, Odisha, and a master's degree from IIT Kharagpur. He has experience in the data and AI domain, including work in data science, and specializes in designing and delivering AI and generative AI solutions by using the IBM Data and AI and IBM watsonx® portfolio, including IBM watsonx.ai®, IBM watsonx®.governance, and IBM watsonx.data. Sailendu has worked on watsonx based generative AI engagements across multiple industries. He has filed patents and published research at the NeurIPS conference. He also participates in hackathons, community initiatives, and activities related to emerging technologies.

Dietmar Fischer is the manager of the IBM Storage Scale AI, Big Data, and Analytics team. He has been with IBM for more than 25 years and has held several positions within the

IBM Storage development organization, including roles in software test, development, project management, and management. Dietmar has a technical background in computer science and works with a team of specialists to develop storage solutions. In previous roles, he founded and served as technical lead of the Cloud Storage Center of Excellence within the IBM Systems EMEA Storage Competence Center, worked as lead architect for an IBM host-based tape diagnostic tool, and contributed as co-inventor to an IBM Storage product. He has also worked on a cloud object-storage research project and has more than 15 years of software-development experience at IBM and other companies. He is a regular speaker at technical conferences and holds patents in storage and networking technology.

Kedar Karmarkar is a development architect with the IBM Storage Scale development team and has contributed to work on protocols, data caching, containerization for IBM Storage Scale, AI solutions, and IBM Storage Scale development-adoption initiatives. Kedar has more than 25 years of experience in infrastructure software and storage development in management and architect roles. Before joining IBM, he led development efforts for network-attached storage (NAS), block-level virtualization and replication systems, and storage-management products. Kedar holds a Bachelor of Engineering (Computer Science) degree from the University of Pune, India.

Mathias Defiebre is an IBM specialist with more than 25 years of experience in data analytics and software-defined storage (SDS). His work focuses on the integration of machine learning with SDS. As part of the IBM EMEA Storage Client Engineering team, he provides pilot and deployment services to clients. He graduated from the University of Cooperative Education Mannheim with a German Diploma in Information Technology Management and a Bachelor of Science degree. Mathias is also a Master Certified IT Specialist and has contributed to several IBM Redbooks® publications.

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

Phillip Gerrard
IBM Redbooks, IBM Infrastructure

Ted Hoover
IBM Product Management

Christina Orosco
IBM Director

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

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

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

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

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

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



Introduction

Artificial intelligence (AI) technologies are developing rapidly. Advances in processing power that are enabled by recent generations of graphics processing units (GPUs) support the development and deployment of increasingly complex AI models. This progress has led to the emergence of large-scale foundation models and large language models (LLMs), which form the basis for technologies such as retrieval-augmented generation (RAG) and agentic AI. Data is central to all of these models. The ability to provide high-quality data through a fast storage solution is important when using costly resources and optimizing IT infrastructure to develop new AI-based solutions. Often, the ability to deliver data quickly determines whether teams can build effective AI solutions rather than relying on limited “AI-like” approaches.

This IBM Redpaper describes IBM watsonx.ai as an AI platform that uses IBM Storage Scale as the underlying high-performance and highly reliable storage solution. The paper describes the architecture of the watsonx.ai and IBM Storage Scale solution and the advantages of this approach. It also provides planning and sizing guidelines and configuration details across the solution stack. Finally, practical applications and use cases demonstrate the solution architecture in typical scenarios.

This chapter describes the following topics:

- ▶ 1.1, “The evolution of storage in the age of artificial intelligence” on page 2
- ▶ 1.2, “Benefits of IBM Storage Scale Global Data Platform” on page 4
- ▶ 1.3, “IBM watsonx ecosystem overview” on page 5

1.1 The evolution of storage in the age of artificial intelligence

AI is becoming a key element across industries and is used to extract value from the large volumes of proprietary data that organizations create and store. As AI adoption increases, so do the associated computing and storage requirements. AI models depend on data throughout their entire lifecycle, from training and testing to deployment and inference. As a result, the role and importance of storage continues to evolve in this new era of AI, as shown in Figure 1-1.

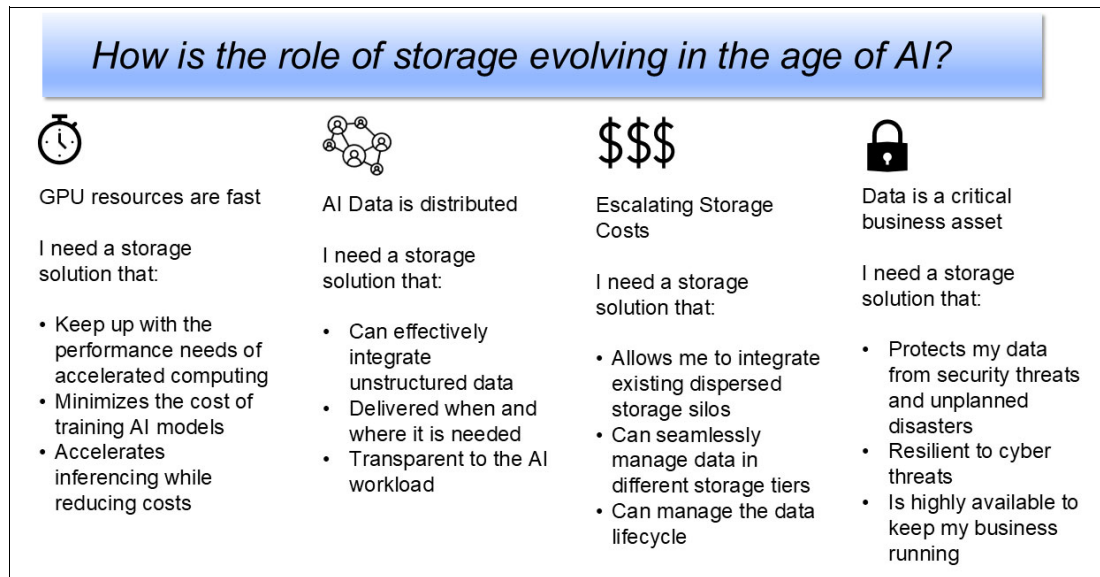


Figure 1-1 The role of storage in the age of AI

In summary, four challenge areas can be identified:

- ▶ **Storage performance.** Modern GPUs are fast, but they rely on storage solutions that can sustain their performance requirements. Slow storage can lead to underused GPUs and inefficient use of resources. High-performance storage helps reduce training time and supports faster inference.
- ▶ **Data distribution and integration.** The data that is required for AI workloads is rarely fully structured or located in one place. Effectively integrating distributed data, including large volumes of unstructured data, is essential to help ensure that it is delivered to the workload at the required time and location. This process must also remain transparent to the AI workload.
- ▶ **Storage efficiency and cost management.** Storage costs can increase quickly if data is duplicated, moved, or copied excessively. A well-designed storage solution must integrate with distributed storage environments, manage data across cost-optimized tiers, and support overall data-lifecycle management.
- ▶ **Data protection and resilience.** Data is a critical business asset that must be protected from security threats and disasters, which includes implementing cyber resilience and disaster recovery processes to help ensure that data remains highly available (HA) and supports continued business operations.

Traditional storage environments typically struggle with at least one of these challenges. As illustrated in Figure 1-2, the main AI phases that lead from data to decision often involve multiple data moves and copies. These phases include the following items:

- ▶ Data collection. Data that is gathered from organizational units within the company is stored in the corporate data lake, which is typically S3- or NFS-connected.
- ▶ Data preparation. The data is loaded, processed, and filtered, for example to remove inappropriate or unusable data.
- ▶ Data transfer. To process the data efficiently, it is moved into a high-performance storage environment.
- ▶ Training. The data is loaded from high-performance storage for training and accessed through POSIX or GPUDirect.
- ▶ Evaluation and deployment. During evaluation, frequent checkpoints are written back to storage to preserve intermediate results.

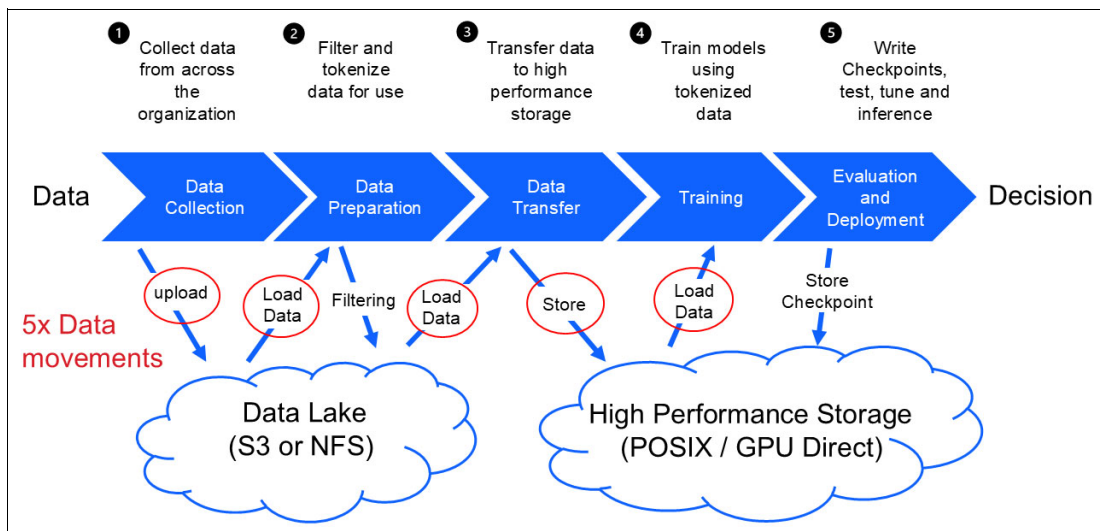


Figure 1-2 Problems of traditional storage approaches

This process does not scale effectively as data volumes continue to grow. Maintaining multiple data copies and managing numerous data-movement workflows becomes increasingly costly and difficult to operate.

To address these challenges, IBM Storage and IBM Storage Scale provide the Global Data Platform, as shown in Figure 1-3. With the Global Data Platform, data movement and duplication are minimized, and the platform provides heterogeneous access mechanisms to data sources such as on-premises network-attached storage (NAS) systems, object stores, and remote cloud object stores. Data is cached locally, supported by IBM Storage Scale System, and tiered to tape when required.

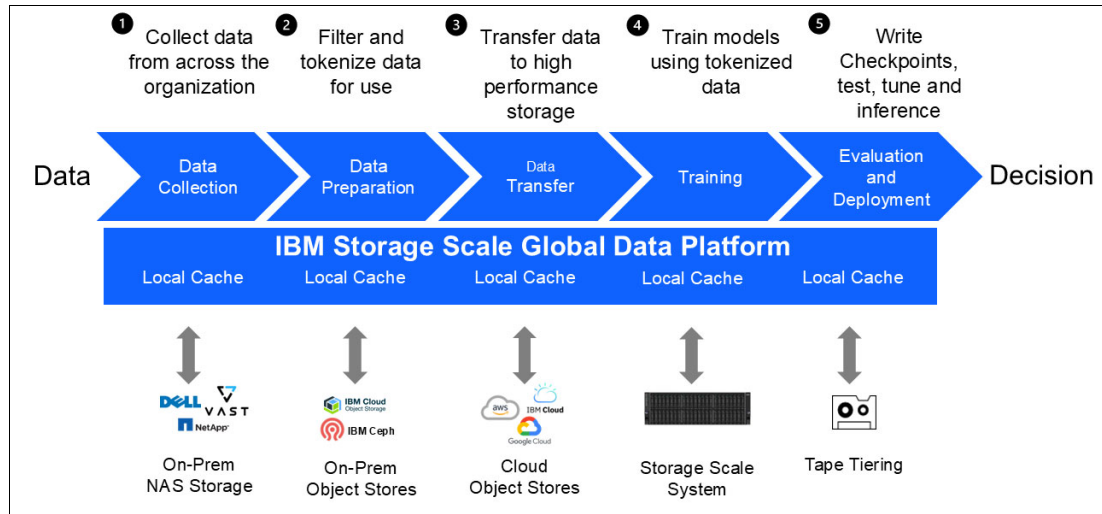


Figure 1-3 IBM Storage Scale Global Data Platform

1.2 Benefits of IBM Storage Scale Global Data Platform

The IBM Storage Scale Global Data Platform offers multiple advantages to address today's storage challenges. At its core is IBM Storage Scale, an enterprise-grade parallel file system that provides superior resiliency, scalability, and control. IBM Storage Scale delivers scalable capacity and performance to handle demanding data analytics, AI and machine learning, content repository, and technical computing workloads.

The IBM Storage Scale Global Data Platform provides capabilities that address storage challenges. At its core is IBM Storage Scale, an enterprise parallel file system that offers resiliency, scalability, and administrative control. IBM Storage Scale supports scalable capacity and performance for data analytics, AI and machine learning, content-repository, and technical-computing workloads.

IBM Storage Scale includes the following features:

- ▶ Scalable performance and capacity
- ▶ Data caching and acceleration
- ▶ Multi-protocol support
- ▶ Update and deployment automation
- ▶ Data resiliency and privacy
- ▶ Intuitive management GUI, APIs, and CLI
- ▶ Container-native support and Container Storage Interface (CSI)
- ▶ Quotas, quality of service, snapshots, and tiering
- ▶ Audit logging and tracking

IBM Storage Scale also includes technologies that support AI use cases, such as [Content-Aware Storage \(CAS\)](#). CAS uses IBM Storage Scale as AI-optimized storage to build AI data pipelines by using AI microservices, such as ones that are based on NVIDIA NIMs, specialized vector databases, and hardware accelerators such as GPUs. IBM Storage Scale can also function as a specialized cache for [generative AI use cases through Key/Value \(KV\) caching](#). KV caching improves LLM inferencing performance by caching intermediate results, which can reduce latency and increase throughput.

1.3 IBM watsonx ecosystem overview

IBM watsonx is integrated into a broad ecosystem, a subset of which is introduced in the following text. As illustrated in Figure 1-4, the deployment infrastructure forms the foundation and can run on-premises or in the cloud. watsonx is built on Red Hat OpenShift as the hybrid-cloud foundation, with Red Hat OpenShift AI providing AI models and tools. The foundation also includes IBM Software Hub, a cloud-native solution that is used to install, manage, and monitor IBM solutions on Red Hat OpenShift.



Figure 1-4 Solution ecosystem overview

watsonx is a portfolio of AI products that includes IBM watsonx.ai, IBM watsonx.governance, and IBM watsonx.data. It is further extended with solutions such as IBM watsonx Orchestrate® for building AI assistants and agents and IBM watsonx Code Assistant® for code generation and software-development tasks. In addition to IBM watsonx, IBM Cloud® Pak for Data provides a modular set of integrated software components for data analysis, organization, and management. IBM Data Product Hub is a solution that is built around data products and enables users to create, publish, and share data products across teams.



Solution architecture

A well-designed solution supports business value through efficient and effective operation, but a poorly designed solution can introduce unanticipated challenges and costs. High-level architecture planning provides the foundation for addressing solution requirements.

This chapter begins with a description of use cases for IBM Storage Scale and IBM watsonx.ai. It then presents the solution architecture and variations that can be adapted to specific requirements.

This chapter describes the following topics:

- ▶ 2.1, “Use cases for IBM Storage Scale with watsonx.ai” on page 8
- ▶ 2.2, “Solution architecture” on page 8
- ▶ 2.3, “Architecture variations” on page 10
- ▶ 2.4, “Benefits of IBM Spectrum Scale for AI use cases” on page 12
- ▶ 2.5, “IBM Storage Scale and IBM watsonx.data” on page 13

2.1 Use cases for IBM Storage Scale with watsonx.ai

The solution that is described in this section provides an artificial intelligence (AI) platform with several use cases, which are outlined in the following items:

- ▶ **Modern AI infrastructure and platform.** IBM watsonx and IBM Storage Scale provide a modern AI infrastructure. Both are designed to address enterprise requirements for performance and scalability. Their design supports hybrid-cloud environments by enabling data caching and improving data-intensive operations. watsonx.ai provides a suite of AI tools and capabilities that support building, training, and deploying custom AI models, accessing prebuilt models, and integrating AI into applications and workflows. IBM Storage Scale is a high-performance and highly scalable storage system that includes an ecosystem of tools and features.
- ▶ **Disaggregated compute and storage infrastructure.** The design of watsonx and IBM Storage Scale follows the concept of disaggregated compute and storage. Unlike monolithic architectures, this approach enables compute and storage components to be built, scaled, and adapted independently. This flexibility supports tailoring environments to specific use cases and adjusting them as requirements evolve.
- ▶ **Modular extension to existing IBM Storage Scale environments.** For customers with existing IBM Storage Scale environments, the solution provides a modular extension that builds on existing investments without requiring costly data copies or data movement to new formats or sources. IBM Storage Scale can serve as the data lakehouse for AI operations without requiring extra third-party components.

2.2 Solution architecture

The architecture of the solution, as illustrated in Figure 2-1 on page 9, follows a layered design and is based on a compute infrastructure and a storage infrastructure. The compute infrastructure hosts Red Hat OpenShift running watsonx.ai and IBM watsonx.data. watsonx.ai manages and runs AI models, workflows, and applications and connects to IBM watsonx.data for data access. IBM watsonx.data acts as an abstraction layer to the storage infrastructure. It hosts vector databases, such as Milvus, backed by local buckets, and provides abstraction through data formats such as Apache Iceberg and access to traditional databases. IBM watsonx.data also supports data preparation by using Apache Spark.

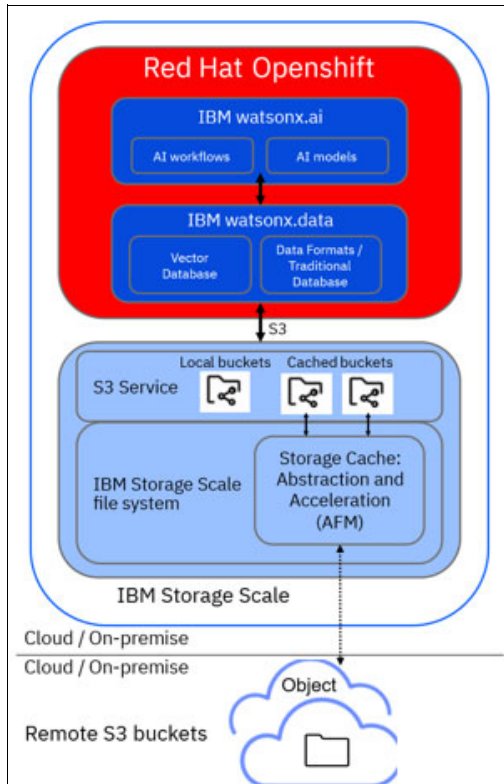


Figure 2-1 Solution architecture

For scenarios that require more specialized data-transformation capabilities, IBM DataStage® can be deployed in addition to the core components. The storage back end is IBM Storage Scale, which provides services for data access through multiple protocols, such as S3 and POSIX. IBM watsonx.data connects to object buckets that are exposed by IBM Storage Scale and accesses the IBM Storage Scale file system by using the S3 protocol.

IBM Storage Scale can also act as an abstraction and caching layer for other cloud and on-premises data sources by using Active File Management (AFM). AFM connects to external data sources to pull data for caching and push updates. This approach supports unified access to distributed data sources and can improve data-processing performance.

This setup also supports hybrid-cloud deployment models and is useful when AI applications run on-premises while most data is in the cloud. It helps improve performance and reduce egress costs by caching data on IBM Storage Scale and exposing the cached buckets to applications without repeatedly retrieving data from the cloud.

2.3 Architecture variations

Depending on the use case, the solution architecture can be adapted. The following sections describe variations of the architecture, which differ primarily in the data-access path. Flexibility in the compute infrastructure is illustrated in Figure 2-2. For example, watsonx.ai can be used for data preparation and can access the IBM Storage Scale file system directly through S3 or through IBM watsonx.data. The AI models within watsonx.ai can be used in AI workflows and can support both training and inferencing scenarios.

IBM watsonx.data, as a higher-level layer, provides abstractions through data formats such as Apache Iceberg and through connections to databases, including vector databases and relational databases.

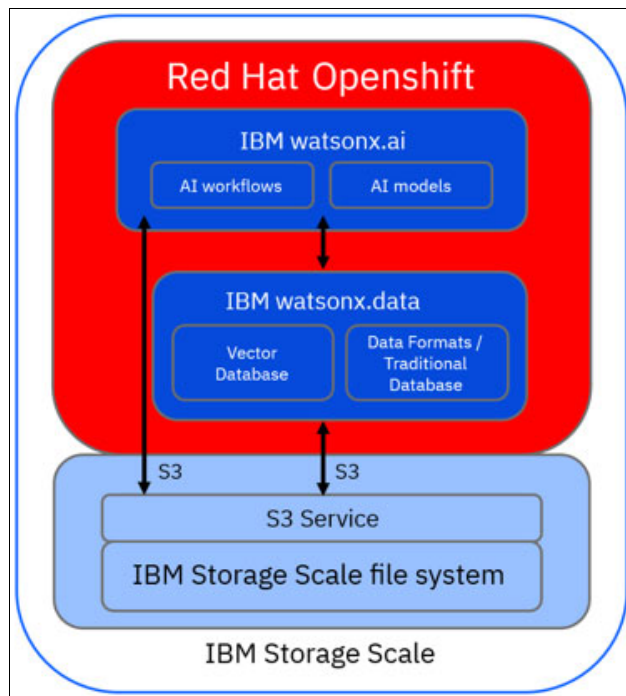


Figure 2-2 Solution architecture variation

If the abstractions that IBM watsonx.data provides are not required, it can be omitted, as shown in Figure 2-3 on page 11. watsonx.ai can access data on IBM Storage Scale by using the cloud-native integration IBM Storage Scale Container Native Storage Access (CNSA), which enables IBM Storage Scale as local storage and integrates the IBM Storage Scale file system through a remote mount. This approach supports both data access and cluster start because required installation data can be kept on the shared storage file system rather than copied.

There are several benefits to using IBM Storage Scale for configuring local storage for the cluster:

- ▶ Organizations do not need to invest in storage-rich servers solely to meet local-storage requirements for installed services. The same centralized storage system, IBM Storage Scale, can be used for both application data and local-storage needs.
- ▶ Many generative AI models have a large storage footprint, and storage demand can increase when models are trained or fine-tuned repeatedly. Using IBM Storage Scale as centralized shared storage supports these requirements.
- ▶ Large models can load more quickly when they are accessed from high-performance, parallel storage such as IBM Storage Scale.

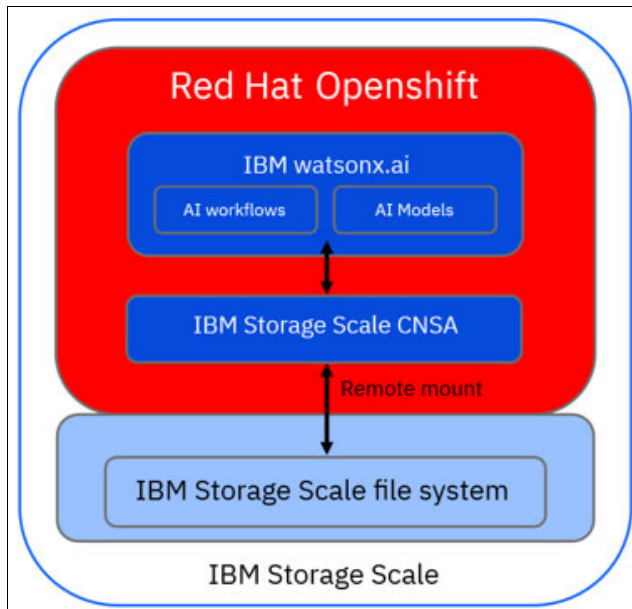


Figure 2-3 Solution architecture variation

Finally, accessing data directly on IBM Storage Scale by using the S3 protocol is an option. As shown in Figure 2-4, watsonx.ai can provide AI workflows that access data on IBM Spectrum® Scale through S3. The data can also be used as input for AI models. This scenario is included for completeness and applies to use cases in which watsonx.ai can operate directly on S3 data or when the abstractions that are provided by IBM watsonx.data are not required. For most use cases, the architectures that were described previously in this chapter are more applicable.

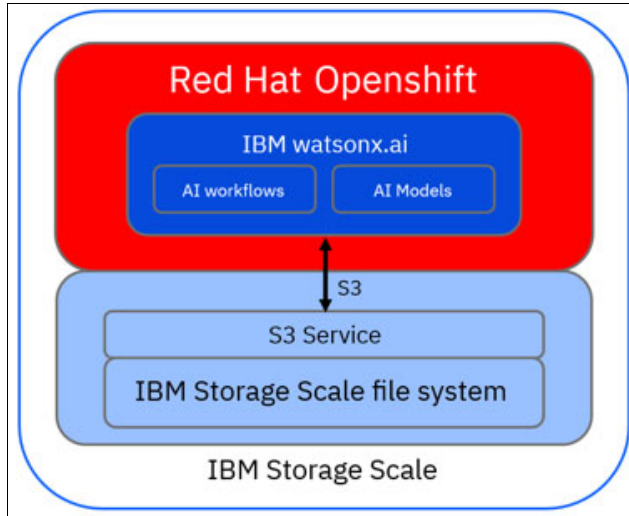


Figure 2-4 Solution architecture variation

2.4 Benefits of IBM Spectrum Scale for AI use cases

Why should a customer consider IBM Storage Scale for AI workloads? The advantages span several areas:

- ▶ **Economics.** IBM Storage Scale provides competitive price-performance characteristics. AI data centers often include large deployments of GPUs, which are costly resources. If the storage back end cannot supply data at the required throughput, GPUs can remain idle. IBM Storage Scale is designed for high-performance data access, which helps support the usage of AI hardware.
- ▶ **Shared data access.** IBM Storage Scale enables simultaneous read/write access to the same data through multiple protocols, such as S3 and POSIX. This capability allows data to be accessed according to workload requirements and helps avoid unnecessary data-copy or data-movement processes that can lead to data silos.
- ▶ **Caching and tiering.** IBM Storage Scale supports caching of data from remote sources by using mechanisms such as AFM. Data can be stored where it is needed and tiered on demand to cost-optimized storage tiers, such as flash or tape.
- ▶ **Enterprise capabilities.** IBM Storage Scale includes features for data availability, reliability, and security. For example, encryption in transit and at rest are built-in security functions. Fault-tolerant design, redundancy, and related capabilities support HA.

2.5 IBM Storage Scale and IBM watsonx.data

As described in this chapter, IBM Storage Scale integrates with IBM watsonx.data. Although IBM watsonx.data is not a prerequisite for watsonx.ai, as shown in the solution-architecture variations, it provides extra features and an abstraction layer on top of IBM Storage Scale. For example, it offers a vector-database service for unstructured data search to support enhanced AI use cases. Selected features of IBM watsonx.data and configuration options for IBM Storage Scale are described in the remainder of this paper. For more information about IBM Storage, see *Accelerating AI and Analytics with IBM watsonx.data and IBM Storage Scale*, REDP-5743.



Planning and sizing

Deployments in the field might begin as proofs of concept (PoCs) and expand to meet workload requirements as environments move into production. Production environments typically grow further as workloads increase over time.

This chapter describes the hardware, network, and software requirements. It provides guidelines and estimates to support planning for the intended environment.

This chapter describes the following topics:

- ▶ 3.1, “Hardware requirements” on page 16
- ▶ 3.2, “Network requirements” on page 17
- ▶ 3.3, “Software requirements” on page 17
- ▶ 3.4, “Sizing guidelines” on page 18

3.1 Hardware requirements

For the overall solution, the hardware requirements can be grouped into two parts: the compute cluster and the storage cluster. In addition, it is important to distinguish between PoC or testing environments and production environments because high availability (HA) might not be required for simple experiments or testing. In contrast, production environments must include redundant hardware, nodes, and components to tolerate hardware failures.

3.1.1 Compute cluster

For Red Hat OpenShift, experiments and initial testing can be performed on a single node, including one graphics processing unit (GPU) for foundation models. A minimal HA environment consists of three control plane nodes and two worker nodes, each with one GPU. The number of worker nodes can be scaled based on workload requirements. Control plane nodes and worker nodes can be deployed on the same servers, so a minimal setup consists of three physical nodes.

- ▶ Control plane nodes. Minimum three nodes, four CPUs, 16 GB of RAM, and 100 GB of storage
- ▶ Worker nodes. Minimum two nodes, two CPUs, 8 GB of RAM (often, foundation models require 96 GB RAM or more), 100 GB of storage, and at least one GPU with 48 GB of RAM (100 GB of RAM recommended)

For more information, see [Red Hat Documentation](#).

The memory, storage, and GPU requirements for the IBM watsonx.ai foundation models are listed in [IBM Documentation](#).

3.1.2 Storage cluster

The storage cluster is composed of the following parts:

- ▶ IBM Storage Scale core. The core of the storage cluster is IBM Storage Scale. It can run as a software-defined storage (SDS) cluster or as an IBM Storage Scale System appliance. The IBM Storage Scale System includes inherent redundancy, with two I/O nodes.
- ▶ AFM gateway nodes for caching and acceleration. At least one node is required to act as an Active File Management (AFM) gateway. This node requires at least 128 GB RAM, and the specific CPU and memory requirements depend on the number of files and file sets that is assigned to AFM. For HA, at least two nodes must serve the AFM gateway role. For more information, see [Planning for AFM](#).
- ▶ Protocol cluster for S3 access. To expose IBM Storage Scale through the S3 protocol, a dedicated protocol cluster is used. The S3 protocol is integrated into the Cluster Export Services (CES), which also enable other protocol access, such as NFS, for IBM Storage Scale. The protocol cluster can consist of a single node, although a 3-node cluster is the minimum configuration for HA. Workloads can be distributed across protocol nodes for load-balancing. For more information about S3 support for IBM Storage Scale, see the [S3 support overview](#).

3.2 Network requirements

In the layered architecture, the network design spans two directions: intra-layer traffic among nodes within the compute and storage clusters, and inter-layer traffic between the compute and storage clusters.

- ▶ Intra-layer network. Communication within the compute cluster or the storage cluster is primarily sensitive to latency. The infrastructure must maintain consistency by checking states and updating metadata when changes occur. Because IBM Storage Scale is a distributed file system, this requirement is especially important on the storage layer where communication delays can directly affect data throughput.
- ▶ Inter-layer network. The network that connects the compute and storage layers serves as the primary data path and is the main focus for bandwidth considerations. Data is stored or cached from remote locations on the storage servers of the IBM Storage Scale file system. Then, the data passes through the protocol nodes to the compute cluster, where it is processed, and the results might be written back through the protocol nodes to the storage servers.

For more information about network design and the IBM Storage Scale System, see [Networking preparation](#).

For the compute layer, a minimum throughput rate of 350 MBps between cluster nodes is required. For more information, see [Network requirements](#).

3.3 Software requirements

The main software components and their requirements are as follows:

- ▶ IBM watsonx.ai

watsonx.ai offers two installation modes: full service and lightweight engine. The full-service mode installs all watsonx.ai services, and the lightweight engine focuses on services for programmatic inferencing of foundation models. Both installation modes require Red Hat OpenShift Container Platform, Red Hat OpenShift AI, and IBM Software Hub. For more information, see [Choosing an IBM watsonx.ai installation mode](#).

- ▶ IBM watsonx.data

IBM watsonx.data requirements are a subset of watsonx.ai and depend on Red Hat OpenShift Container Platform and IBM Software Hub. Both watsonx.ai and IBM watsonx.data can run in the same Red Hat OpenShift cluster. For more information about installation details for IBM watsonx.data, see [Installing watsonx.data](#).

- ▶ IBM Storage Scale – S3 protocol support

The S3 protocol is integrated into CES and can be configured and managed in a manner similar to other CES-managed protocols, such as NFS. For more information about installation and setup details for S3 support in IBM Storage Scale, see [Deploying IBM Storage Scale CES-S3 using Ansible Toolkit - 5.2.1](#).

- ▶ IBM Storage Scale – server cluster

The primary storage cluster is provided by IBM Storage Scale. It establishes a distributed and fault-tolerant file system with features for security, availability, and observability. For more information about instructions for setting up and installing IBM Storage Scale, see [Installing IBM Storage Scale and creating a cluster](#).

- ▶ IBM Storage Scale AFM

AFM provides caching for external data sources. It can cache data from on-premises storage platforms or remote storage systems such as public clouds. For more information about installation details, see [Installation of Active File Management \(AFM\)](#).

3.4 Sizing guidelines

This section shows recommended sizing for the components. For more information, see the relevant product documentation.

3.4.1 watsonx.ai

Table 3-1 shows the best practice sizings for an watsonx.ai on x86 architecture. For more information about the resource requirements for individual services (such as IBM Watson® Studio), see [Hardware requirements](#).

Table 3-1 watsonx.ai system requirements overview

Node role	Number of servers	Number of vCPUs	Memory	Storage
Control plane	3	4 vCPUs per node	16 GB RAM per node	N/A
Infra	3	4 vCPUs per node	24 GB RAM per node	N/A
Worker	3	16 vCPUs per node	128 GB RAM per node	500 GB per node
Load Balancer	2	2 vCPUs per node	8 GB RAM per node	100 GB

3.4.2 IBM watsonx.data

Table 3-2 shows the best practice sizings for IBM watsonx.data. For more information, see [Hardware requirements](#).

Table 3-2 *watsonx.data system requirements overview*

Node role	Number of servers	Number of vCPUs	Memory	Storage
Primary + Infra	3 (both roles on the same node)	8 vCPU per node	32 GB RAM per node	N/A
Worker	3	16 vCPU per node	128 GB RAM per node	300 GB per node
Load Balancer	2	4 vCPU per node	8 GB RAM per node	100 GB

3.4.3 IBM Storage Scale System

shows the best practice sizings and specifications of the IBM Storage Scale System models. To see the full data sheet, see the [IBM Storage for Data & AI Data Sheet](#).

Table 3-3 *IBM Storage Scale Requirements overview*

Category	IBM Storage Scale 3500	IBM Storage Scale 6000
Size	2 rack units (RUs)	4 RUs
Maximum capacity	24 x 30.72 TB NVMe	48 x 30.72 TB NVMe 48 x 38 TB FCM4 modules
Maximum throughput	126 GBps	330 GBps
Expansion	Up to 4 direct-attached JBODs	Up to 9 direct-attached JBODs
Data transfer	12 Gb SAS	24 Gb SAS



Configuring the solution

When deploying the proposed solution, several components and applications require configuration. The configuration details depend on the use case, such as whether an object bucket on the storage layer is defined locally or must function as a locally cached and accelerated bucket from a remote source.

This chapter described the storage configurations that are required for the solution architecture, which include configuring IBM Storage Scale and its file system, exposing S3 buckets for applications, and explaining how IBM watsonx accesses data on the storage layer.

This chapter describes the following topics:

- ▶ 4.1, “Configuring IBM Storage Scale System and IBM Storage Scale” on page 22
- ▶ 4.2, “Configuring IBM Storage Scale CNSA” on page 24
- ▶ 4.3, “IBM Storage Scale CES S3” on page 25
- ▶ 4.4, “Configuring data abstraction and acceleration” on page 29
- ▶ 4.5, “Defining IBM Storage Scale S3 buckets to IBM watsonx.data” on page 32
- ▶ 4.6, “Adding a Milvus vector database service to IBM watsonx.data” on page 34

4.1 Configuring IBM Storage Scale System and IBM Storage Scale

An example deployment that follows the recommended architecture is shown in Figure 4-1. The deployment uses a tiered architecture with a separation of compute and storage infrastructure. The three tiers are identified as follows:

- Compute tier

This tier contains the application layer. It runs the Red Hat OpenShift Container Platform, which hosts `watsonx.ai` and `IBM watsonx.data`. These services use the S3 object protocol to access the storage infrastructure. Alternatively, or in addition, IBM Storage Scale Container Native Storage Access (CNSA) can be used to remotely mount the IBM Storage Scale file system from the IBM Storage Scale server cluster.

- IBM Storage Scale client cluster

This cluster contains the S3 protocol nodes that provide S3 access for IBM `watsonx`. The client cluster that is shown includes multiple protocol nodes to support high availability (HA). An uneven number of nodes is a best practice to achieve quorum, although an even number can be used with a tiebreaker configuration. The nodes in the IBM Storage Scale client cluster remotely mount the file system or file systems of the IBM Storage Scale server cluster.

- IBM Storage Scale server cluster

This cluster consists of I/O server nodes and a node that serves as an Active File Management (AFM) gateway. The server cluster owns the file system and provides an acceleration layer by using AFM to connect with other S3 or NFS data sources or a remote IBM Storage Scale file system.

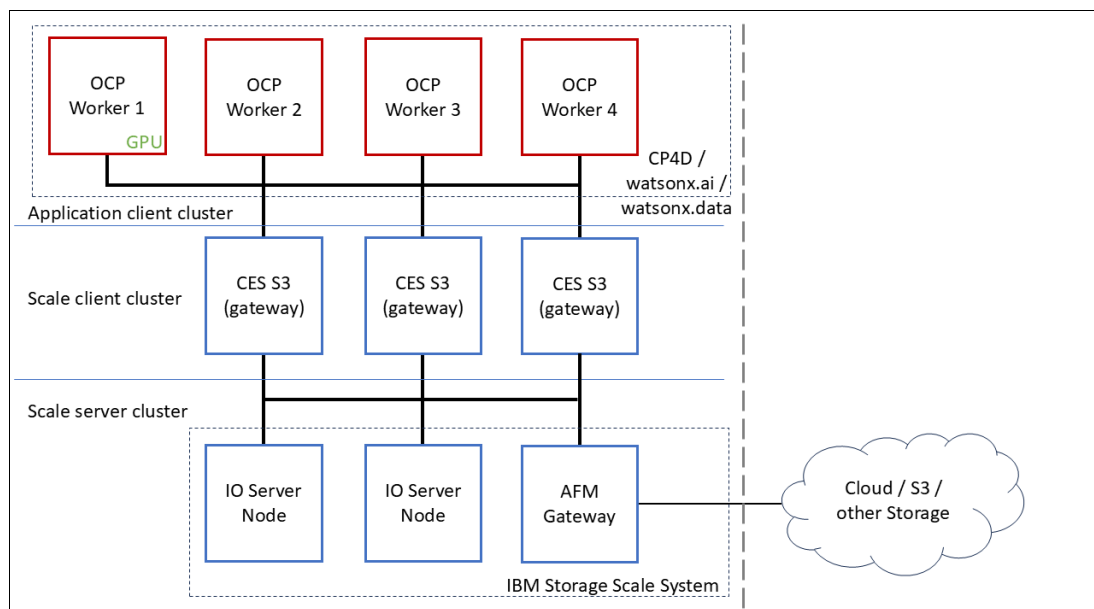


Figure 4-1 Three-tiered deployment for IBM `watsonx.ai` and IBM `watsonx.data` with IBM Storage Scale

The IBM Storage Scale client cluster, which contains the S3 service, is used to configure S3 access for watsonx. This process is described in the following sections. First, an S3 account is required, and it can be created after the corresponding user and group are defined within the operating system. After the account is created, an S3 bucket is defined. The CES IP address and port, account credentials (access key and secret key), and bucket name are required to configure the connection from watsonx.

Example 4-1 shows the configurations of the IBM Storage Scale client cluster that is used as an example for this paper.

Example 4-1 Configurations of the IBM Storage Scale client cluster that is used in this paper

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

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

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

The Cluster Export Services (CES) IP addresses are configured and assigned to the nodes of the IBM Storage Scale client cluster. In Example 4-2, there are two CES IP addresses that are set up.

Example 4-2 Listing the CES IP addresses

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

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

Example 4-3 Listing the remote mounted file systems in the IBM Storage Scale client cluster

```
[root@fscs-sr650-46 ~]# mmremotefs show all
Local Name Remote Name Cluster nameMount PointMount OptionsAutomount Drive
Priority
essDataessDataess3k5.bda.scale.com /gpfs/essDataess3k5rwno-0
essCesRoot
essCesRootess3k5.bda.scale.com /gpfs/essCesRootess3k5rwno-0
```

4.2 Configuring IBM Storage Scale CNSA

Use the IBM Storage Scale CNSA cluster to access the IBM Storage Scale file system. IBM Storage Scale CNSA is configured on the same Red Hat OpenShift container cluster that runs the watsonx services. For more information, see [IBM Storage Scale Container Native](#).

Here are the high-level steps:

1. Prepare the physical IBM Storage Scale cluster for container access, which includes configuring the file system or file systems for remote mounting in the CNSA cluster and creating the users and groups for the container operator and Container Storage Interface (CSI) users.
2. Configure the worker and primary nodes on the Red Hat OpenShift cluster to prepare for the CNSA cluster setup.
3. Deploy the CNSA cluster and the remote-mount file system.

The CNSA cluster remotely mounts the file system or file systems of the IBM Storage Scale server cluster and exposes the storage through Kubernetes custom resources that are called persistent volumes (PVs). Red Hat OpenShift applications consume the PVs through persistent volume claims (PVCs) to meet their persistent-storage requirements.

Example 4-4 shows the CNSA cluster custom resource (CR) artifact and the pods that belong to the cluster.

Example 4-4 Configurations of the Scale CNSA cluster used in this paper

```
$ oc get clusters.scale.spectrum.ibm.com
NAME                EDITION    AGE
ibm-spectrum-scale  data-access 230d
```



```
[root@fscs-sr650-36 ~]# $ oc get pods -n ibm-spectrum-scale
NAME                READY   STATUS    RESTARTS   AGE
compute-x1          2/2     Running   0           109d
compute-x2          2/2     Running   0           109d
compute-x3          2/2     Running   0           109d
compute-x4          2/2     Running   0           109d
ibm-spectrum-scale-gui-0  4/4     Running   0           109d
ibm-spectrum-scale-gui-1  4/4     Running   0           109d
ibm-spectrum-scale-pmcollector-0  2/2     Running   0           109d
ibm-spectrum-scale-pmcollector-1  2/2     Running   0           109d
```



```
$ oc get filesystem -n ibm-spectrum-scale -o wide
NAME                ESTABLISHED  REMOTE CLUSTER      MAINTENANCE MODE  AGE
remote-sample      True         remoteccluster-sample  not supported      229d
```

Example 4-5 on page 25 shows the storage classes. The storage class `ibm-spectrum-scale-internal` is exposed by the CNSA cluster and is used internally by the GUI and `pmcollector` services. The storage class `ibm-spectrum-scale-sample` is used by `watsonx` services and applications.

Example 4-5 Storage classes that are exposed by the IBM Storage Scale CNSA cluster that is used in this paper

```
$ oc get storageclass |grep scale
ibm-spectrum-scale-internal          kubernetes.io/no-provisioner
Delete                               WaitForFirstConsumer             false                             230d
ibm-spectrum-scale-sample           spectrumscale.csi.ibm.com
Delete                               Immediate                         true                             229d
```

4.3 IBM Storage Scale CES S3

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

4.3.1 Installing and configuring the IBM Storage Scale S3 service

To install and enable the S3 service, complete the following steps:

1. To set up and install IBM Storage Scale, use the Installation Toolkit. For more information about the Installation Toolkit, see [Installing](#).
2. After the basic information is set up, define the protocol nodes for the cluster by running the following command:

```
# cd /usr/lpp/mmfs/6.0.0.0/ansible-toolkit/
# ./spectrumscale node add hostname -p ...
```

3. Define the CES IP addresses for the cluster to export by running the following command, where <EXPORT_IP_POOL> is a comma-separated list of IP addresses that are used by applications to access the S3 service:

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

Note: Reverse domain name system (DNS) lookup must be available for all CES IP addresses. The CES IP addresses must be unique and cannot be cluster node IP addresses.

4. Configure the CES shared root file system, which is used for configuration and administration of the CES protocols, by running the following command:

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

Note: As a best practice, keep the CES shared root as a separate file system. The CES shared root must be at least 4 GB.

5. Enable the S3 protocol by running the following command:

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

6. Review the configuration and perform a precheck of the deployment by running the following command:

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

7. Deploy the changes by running the following command:

```
# ./spectrumscale deploy
```

- Verify that the S3 services are running on the designated S3 protocol nodes by running the following command:

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

- To view the S3 protocol configuration, run the following command:

```
# mms3 config list
```

Example 4-6 shows the default configuration of the S3 service.

Example 4-6 Default configuration of the S3 service

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

4.3.2 DNS load balancer for S3

For increased performance, this example uses a load balancer for the IBM Storage Scale S3 service. A simple DNS-based load balancer was selected. A DNS entry is created with all S3 endpoint IP addresses as A records. Clients resolve the DNS entry to one IP address from the list and cache the result during the time-to-live (TTL) value.

DNS load balancers can provide good performance with no additional network hop, unlike options such as a Network Load Balancer (NLB) or Application Load Balancer (ALB). DNS-based load-balancing is also straightforward to maintain and does not require extra hardware. A limitation is that the DNS-based distribution is random and does not account for host contention, which can result in uneven workload distribution.

In this example, a dnsmasq-based DNS load balancer was created by adding DNS A records for the three CES IP addresses and restarting the dnsmasq service:

```
10.10.1.119 cesip.bda.scale.ibm.com
10.10.1.120 cesip.bda.scale.ibm.com
10.10.1.121 cesip.bda.scale.ibm.com
```

The DNS name `cesip.bda.scale.ibm.com` is then used to configure watsonx services to reach the IBM Storage Scale S3 service.

4.3.3 Configuring file sets as backing storage for S3 buckets

A *file set* is a subtree of an IBM Storage Scale file system that, in many respects, behaves like an independent file system. File sets provide a way to partition the file system so that administrative operations can be performed at a finer granularity than at the file-system level.

File sets can have their own defined quotas for data and inodes. The owning file set becomes an attribute of each file and is used to enforce IBM Storage Scale policies such as automated tiering and placement, encryption, and compression. Each file set is mounted at a directory path (called the JunctionPath) within the IBM Storage Scale file system. An S3 bucket can be defined over the mount path.

For more information, see [Filesets](#).

The following scenarios illustrate when it might be preferable to create an S3 bucket over a file set rather than over a regular directory:

- ▶ Assigning a quota to the storage space that a bucket can use.
- ▶ Limiting the total number of objects that a bucket can contain.
- ▶ Automatically encrypting all data (objects) in the bucket on disk.
- ▶ Defining automated tiering and placement policies for a bucket. For example, if cold data is uploaded, it can be moved automatically to a slower storage tier.
- ▶ Creating time-based snapshots at the bucket level. File set snapshots can be created instead of creating a snapshot of the entire file system.

Example 4-7 shows how to create an independent file set.

Example 4-7 Creating an independent file set

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

Example 4-8 shows how to link the file set to an IBM Storage Scale mount path.

Example 4-8 Linking the file set to an IBM Storage Scale mount path

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

Create a S3 bucket over the file set's mount path (directory), as described in 4.3.4, “Creating S3 buckets” on page 27. In Example 4-8, the mount path is `/gpfs/essData/watsonx/`, which is the default bucket path (`--newBucketsPath`) for this S3 account.

4.3.4 Creating S3 buckets

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

Alternatively, a bucket can be configured over a pre-existing directory. For example, a bucket can be configured over the mount-point directory of an IBM Storage Scale file set so that the file set becomes the backing storage for the S3 bucket.

To create a S3 bucket, complete the following steps:

1. Create a S3 account and associate the account to a system user by running the following command:

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

- <uid> and <gid> are the POSIX UID and GID that are associated with the S3 account. These parameters are not needed if an account name is passed. The account name should be a valid system username.
- <Path> is the file system’s absolute path, which acts as a base path for S3 buckets that are created by using the S3 API. This path can be overridden for buckets that are created with the **mms3 bucket create** command.

To view the details that are associated with this S3 account, including its AWS access credentials, run the following command:

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

2. Create one or more S3 buckets that correspond to the S3 account.

There are two ways that you can use to create a bucket:

- Method 1. Run the **mms3** command:

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

- <S3 account-name> is the name of the account that you use for the bucket.
- <Path> is the file system’s absolute path, which includes the directory for the bucket that you use for bucket creation. This bucket may be different than the default bucket path (**--newBucketsPath**) that is configured for the S3 account.

The command creates a directory with system path <Path>, which corresponds to the S3 bucket.

Here is an example of creating an S3 bucket by running the **mms3** command:

- i. Create a S3 account that is named “watsonx” by running the following command:

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

- ii. View the details that are associated with this S3 account including its AWS access credentials by running the following command:

```
# mms3 account list watsonx  
NameNew Buckets PathUid Gid Access KeySecret Key  
watsonx /gpfs/ess3k54/watsonx/ 2002 100 <Our AWS_ACCESS_KEY_ID>< Our  
AWS_SECRET_ACCESS_KEY>
```

- iii. Create a S3 bucket under the default Bucket path by running the following command:

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

- iv. Create a S3 bucket that is not under the default Bucket path by running the following command:

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

- v. Create a bucket over an existing directory with data in it:

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

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

For file set backed storage, the mount paths that you create are owned by the root user. Change the ownership of that directory to the S3 account by running the following command:

```
# chown watsonx:users /gpfs/essData/watsonx/b-wxd-fset1
# mms3 bucket create b-watsonx3 -accountName watsonx -filesystemPath
/gpfs/essData/data-dir1
```

Note: The directory '/gpfs/essData/data-dir1' for bucket exists. Skipping update of ownership and the setting of permissions of the directory for the user with uid:gid=2002:100

Bucket b-watsonx3 created successfully

- Method 2 - Using the S3 API, as shown in Example 4-9.

Example 4-9 Creating an S3 bucket by using the S3 API

```
# wget https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip # unzip
awscli-exe-linux-x86_64.zip

# cd aws
# ./install
# alias s3u2='AWS_ACCESS_KEY_ID=<Your_AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<Your_AWS_SECRET_ACCESS_KEY> aws --endpoint
https://10.11.94.182:6443 s3'

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

4.4 Configuring data abstraction and acceleration

Configure IBM Storage Scale AFM to enable storage abstraction and acceleration for dispersed buckets. This workflow includes the following steps:

1. Create a rule in AFM that defines the connection to the remote S3 bucket. This action creates a file set in IBM Storage Scale that corresponds to the remote S3 bucket.
2. Create an S3 bucket over that file set. This bucket abstracts the remote object bucket and is exposed through the IBM Storage Scale S3 interface.

To configure storage acceleration over remote buckets, complete the following steps:

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

```
# mmchnode --gateway -N <AFM node hostname>
```
2. Get the AWS access key ID and secret key for your remote bucket instance. For example, if you use IBM Cloud Object Storage as a service on IBM Cloud, select **cloud.ibm.com** → **Instances** → **Storage**, click the Service Credentials tab, and click the down arrow. The key details are in `cos_hmac_keys`.

3. Log in to an AFM gateway node.
4. Create the access keys in AFM for the remote object bucket by running the following command:


```
# mmafmcoskeys bucket[:{[Region@]Server|ExportMap}] set {<access key> <secret key> | --keyfile filePath}
```
5. Create an AFM relationship for the remote S3 bucket, as shown in Example 4-10.

Example 4-10 Creating an AFM relationship for a remote S3 bucket

```
# mmafmconfig <Device> <FilesetName> --endpoint
http[s]://{[Region@]Server|ExportMap}[:port]--object-fs--bucket
<BucketName>--mode <AccessMode> --dir <Path> --debug
```

- <Device> is the name of your IBM Storage Scale file system.
- <FilesetName> is the name of the file set that you want to create for the remote S3 object.
- <BucketName> is the name of the remote S3 bucket.
- <mode> is the AFM Access Mode.
- <Path> is the relative directory path under the file system mount directory, where you mount the file set (LinkPath).

Note: The `--dir` parameter passes to the command to help ensure that the file set is created under the S3 New Buckets Path (from the `mms3 account list` command).

To see the newly created file set, run the following command:

```
# mmlsfileset <Device>
```

To see the relationship of the file set with the remote bucket, run the following command:

```
# mmafmctl <Device> getstate
```

<Device> is the name of the Storage Scale file system.

6. Create a S3 bucket over the file set's mount path. Change the ownership of the directory to the account of the S3 bucket.

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

7. Create the S3 bucket pointing to that directory:

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

Configuring data abstraction and acceleration example

In this example, there is a remote S3 bucket that is named `chm-cos-s3-bucket` on IBM Cloud Object Storage. To create a virtual and accelerated S3 bucket that is named `b-watsonx` for the IBM Cloud Object Storage bucket, complete the following steps:

1. Designate an AFM node by running the following command:

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

2. Run the following commands on the AFM node:

- Create access keys in AFM for the IBM Cloud Object Storage object:

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

- View the AFM access key by running the following command:

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

3. Create an AFM relationship for the IBM Cloud Object Storage bucket by running the following command:

```
# mmafmcosconfig essData ibmcos-bucket \
--endpoint http://s3.us-east.cloud-object-storage.appdomain.cloud \
--object-fs \
--bucket chm-cos-s3-bucket \
--mode iw \
--dir watsonx/ibmcos-bucket --debug
```

Note: The value of the `-dir` parameter is chosen because the `essData` file system is mounted at `/gpfs/essData`, and the default path for S3 buckets (as shown by the `mms3 account list` command) is `/gpfs/essData/watsonx`. This configuration results in the `ibmcos-bucket` file set being mounted at the relative path `watsonx/ibmcos-bucket` under the file system mount point. A corresponding S3 bucket is created at this location later.

The command produces the output that is shown in Example 4-11.

Example 4-11 Output of the mmafmcosconfig command

```
# mmafmcosconfig essData ibmcos-bucket \
--endpoint http://s3.us-east.cloud-object-storage.appdomain.cloud \
--object-fs \
--bucket chm-cos-s3-bucket \
--mode iw \
--dir watsonx/ibmcos-bucket --debug

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

4. To view the newly created AFM relationship, run the `mmafctl` command, as shown in Example 4-12.

Example 4-12 mmafctl output

```
# mmafctl essData getstate
Fileset Name
Fileset Target                               Cache State
Gateway Node           Queue Length           Queue numExec

ibmcos-bucket
http://s3.us-east.cloud-object-storage.appdomain.cloud:80/chm-cos-s3-bucket Active
ems3000.bda.scale.ibm.com           0                               3
```

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

5. Create a S3 bucket on the directory `/gpfs/essData/watsonx/ibmcos-bucket`, as shown in Example 4-13.

Example 4-13 S3 bucket creation

```
# chown watsonx:users ibmcos-bucket/

# mms3 bucket create b-watsonx-cos --accountName watsonx --filesystemPath
/gpfs/essData/watsonx/ibmcos-bucket
Starting to create bucket with name b-watsonx-cos
```

Note: The directory `'/gpfs/essData/watsonx/ibmcos-bucket'` for bucket exists.
Skipping update of ownership and the setting of permissions of the directory for the user with `uid:gid=2002:100`
Bucket `b-watsonx-cos` created successfully

4.5 Defining IBM Storage Scale S3 buckets to IBM watsonx.data

You can install IBM watsonx.data by using the procedure that is described at [Installing watsonx.data](#).

To register a Storage Scale S3 bucket to your IBM watsonx.data instance as externally managed storage and associate a catalog for the bucket, complete the following steps. This catalog serves as the query interface for IBM watsonx.data for the data that is stored within the bucket, as shown in Figure 4-2 on page 33.

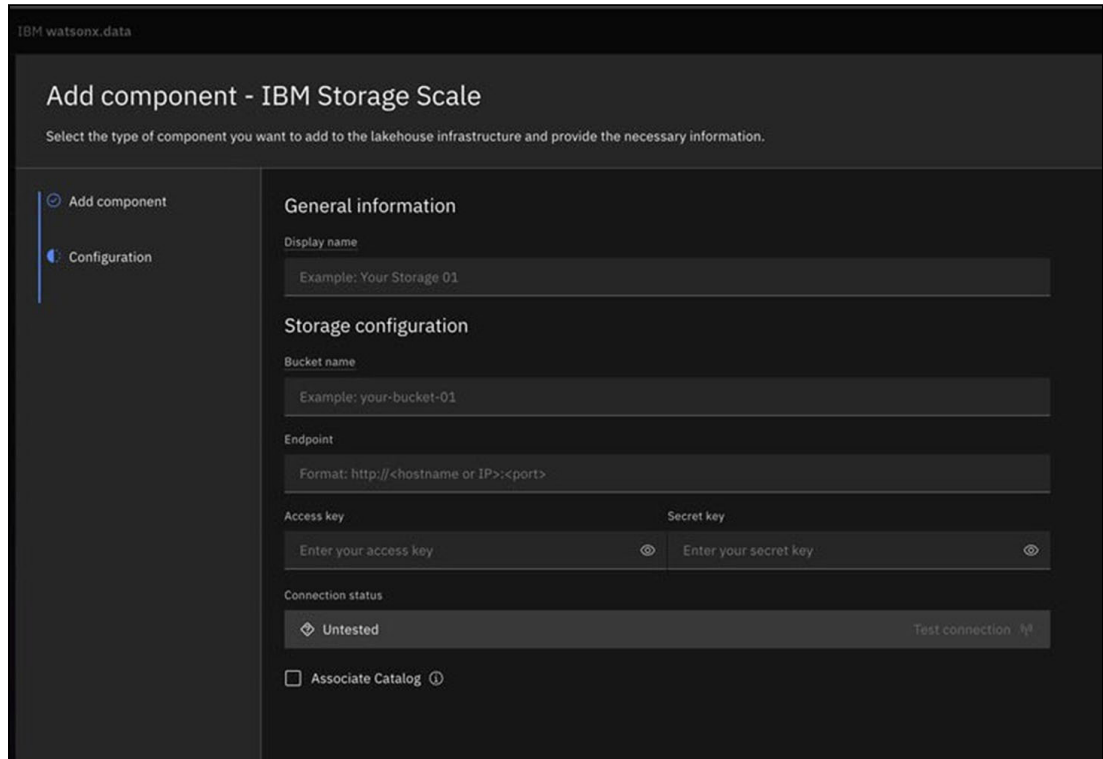


Figure 4-2 IBM watsonX.data window for adding IBM Storage Scale component

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

Note: For higher throughput and performance from the S3 service, you can use a load balancer to distribute the workload across all protocol nodes. If a load balancer is configured, use the DNS name of the load balancer in the endpoint URL rather than using a CES IP address directly. For more information about using a load balancer, see [Load balancing](#).

- Access key. Enter your access key.
- Secret key. Enter your secret key.

- Connection Status. Click **Test connection** to test the bucket connection.
- Associate Catalog. Select the checkbox to add a catalog for your storage.
- Catalog type. Select the catalog type from the list. The recommended catalog is Apache Iceberg. The other options for catalog are Apache Hive, Apache Hudi, and Delta Lake.
- Catalog name. Choose a name for the associated catalog.

To add a bucket-catalog pair, see [Adding a storage-catalog pair](#).

5. Create an engine instance, such as Presto, and associate the catalog with that engine. This configuration makes the S3 bucket discoverable through the catalog. You can then create schemas and tables under the storage catalog, as shown in the following command:

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

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

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

This command creates a directory that is called schema1 under the bucket's file system path in IBM Storage Scale. Data for all tables that are created under this schema are in this directory.

4.6 Adding a Milvus vector database service to IBM watsonx.data

The Milvus service exposes Google Remote Procedure Call (gRPC) and REST API endpoints along with an SSL public certificate. Applications from watsonx.ai connect to these endpoints to ingest data and query the vector database.

To add a Milvus vector-database service instance, complete the following steps:

1. Open the IBM watsonx.data Infrastructure view window, as shown in Figure 4-3 on page 35.
2. Select the size of the database based on infrastructure requirements.

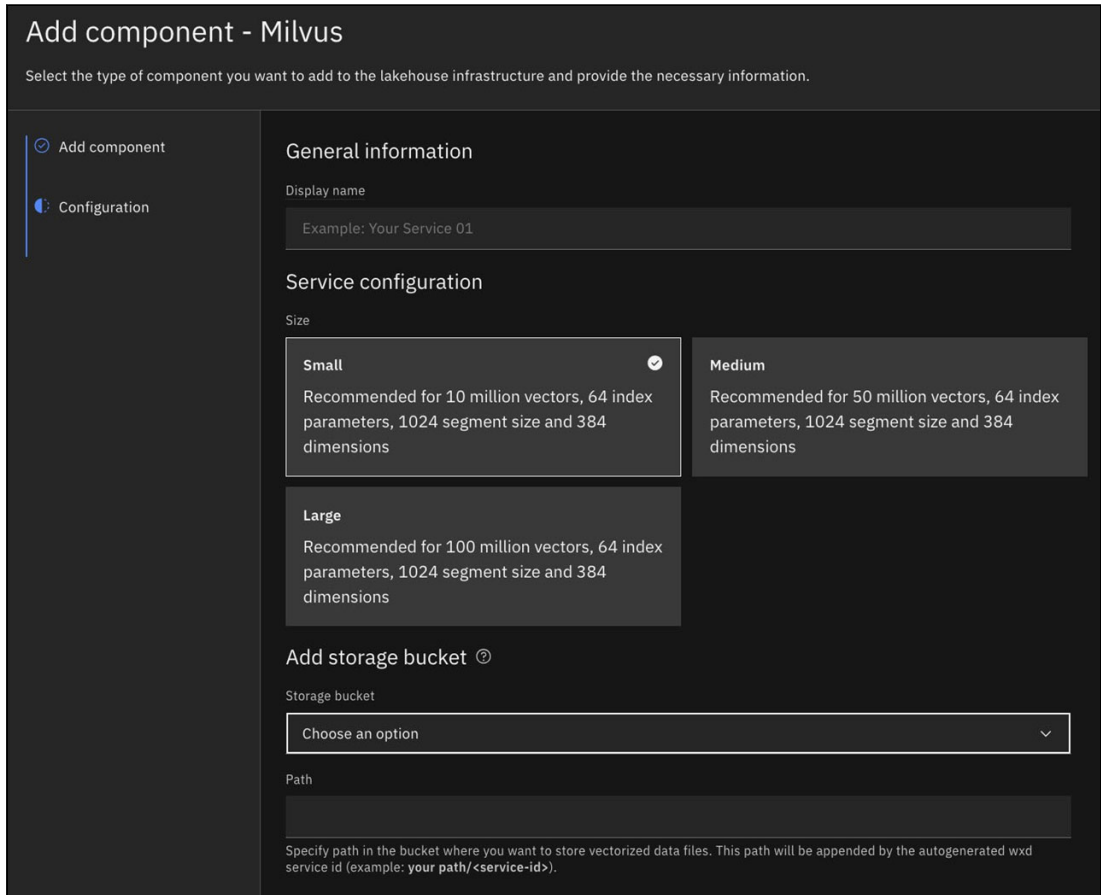


Figure 4-3 IBM watsonx.data GUI view of the analytics infrastructure

3. Choose one of the IBM Storage Scale buckets to use as the backing storage for the Milvus vector database.
4. Choose a path under the IBM Storage Scale S3 bucket (such as `/milvus01_data`) where the vector embeddings will be stored. A common bucket can be used for storing vector embeddings and other analytics data such as Parquet files. Alternatively, a dedicated IBM Storage Scale bucket can be used for Milvus.

Figure 4-4 shows a sample view of the IBM watsonx.data Infrastructure Manager GUI, including multiple engines, services, catalogs, and IBM Storage Scale buckets that are defined in the environment. It illustrates the separation of compute, data, and metadata layers and shows the disaggregated architecture with modular components.

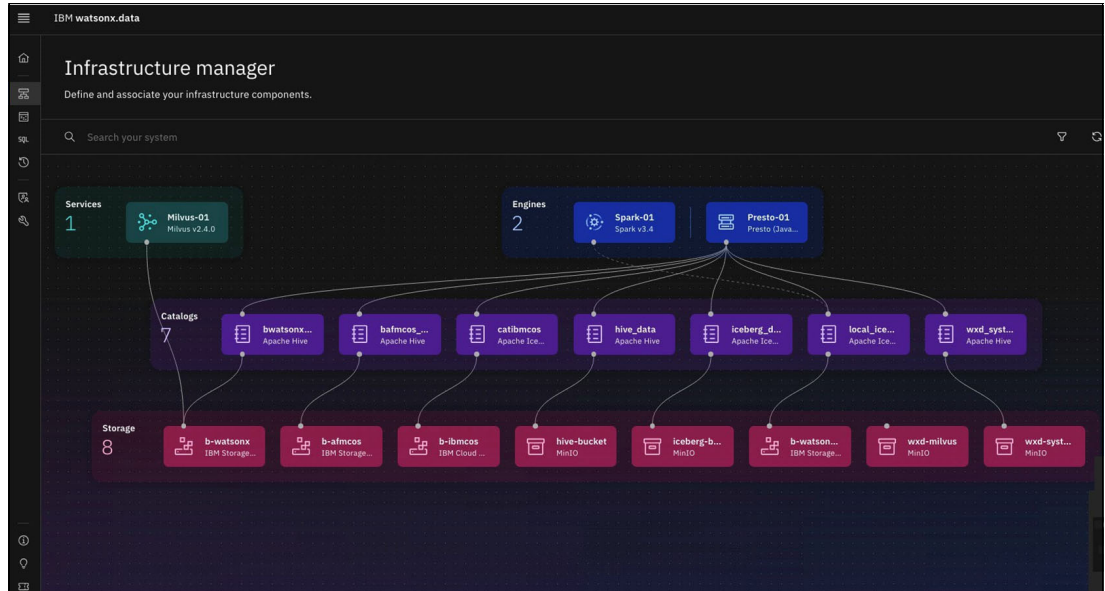


Figure 4-4 Sample view of the IBM watsonx.data Infrastructure manager GUI



Examples, use cases, and solution application

Artificial intelligence (AI) use cases are broad, and models are evolving rapidly. Large language models (LLMs) enable scenarios that were not feasible only a few years ago. One challenge of LLMs is that they can generate inaccurate or invented responses. At the time of this writing, retrieval-augmented generation (RAG) is a commonly used technique to help address this issue.

This chapter describes text-extraction and RAG use cases and shows how they are implemented in the solution architecture. Open-source tools and benchmarks help demonstrate the approach. This chapter also describes an extended concept, such as a role-based content filter, and how it can be incorporated into the RAG flow.

This chapter describes the following topics:

- ▶ 5.1, “Working with IBM watsonx.ai” on page 38
- ▶ 5.2, “Use case: Robust entity extraction and summarization with Docling” on page 40
- ▶ 5.3, “Use case: Bulk data ingest and query evaluation with Open RAG Benchmark” on page 45
- ▶ 5.4, “Use case: RAG with a role-based content filter on IBM Storage Scale” on page 51

5.1 Working with IBM watsonx.ai

This section describes ways to interact and work with watsonx.ai. This section demonstrates a simple chat workflow that uses custom data to answer questions. This workflow is the main concept of RAG. This example uses the environment setup that is described in Chapter 4, “Configuring the solution” on page 21.

5.1.1 GUI for an interactive workflow

watsonx.ai includes a built-in chat interface and workbench that is called Prompt Lab, as shown in Figure 5-1. It provides flexibility for working with foundation models by using chat prompts, applying prompt-engineering techniques, and scanning custom data to support prompt creation.

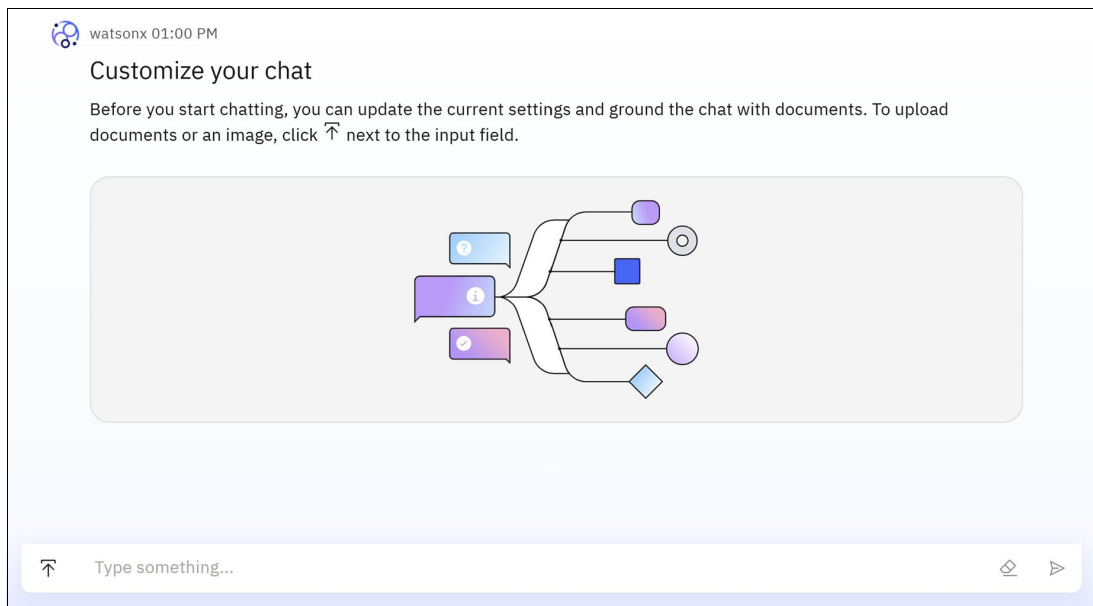


Figure 5-1 Prompt Lab UI

Foundation models can scan and analyze custom data. The Prompt Lab UI enables users to upload documents as needed to support answering a question, as shown in Figure 5-2 on page 39.

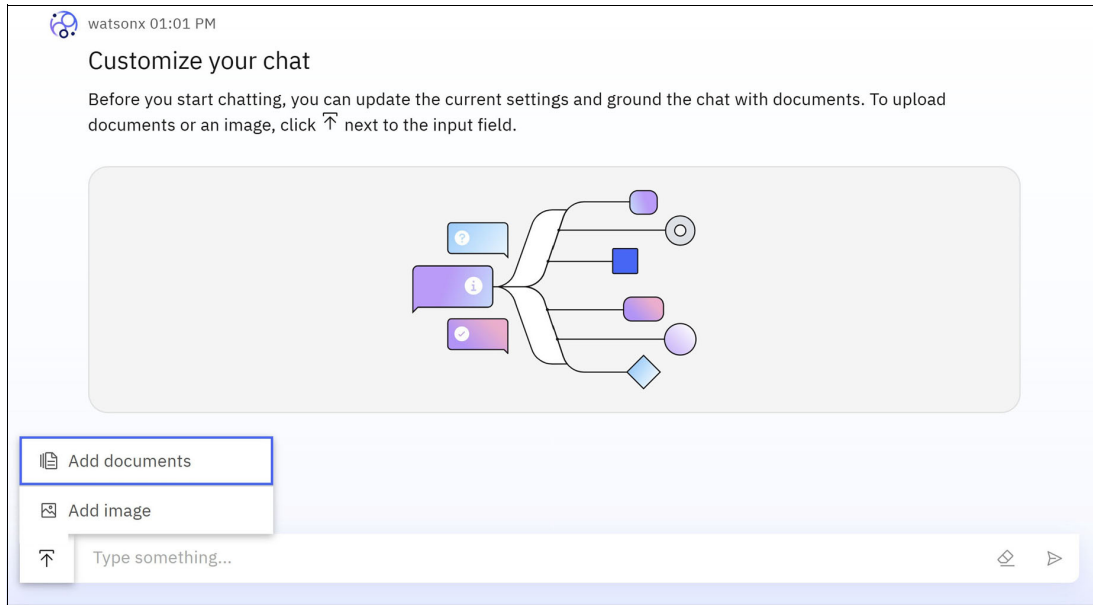


Figure 5-2 Add documents to Prompt Lab

The document being analyzed is stored in memory or can be persistently stored in IBM watsonx.data by default, as shown in Figure 5-3.

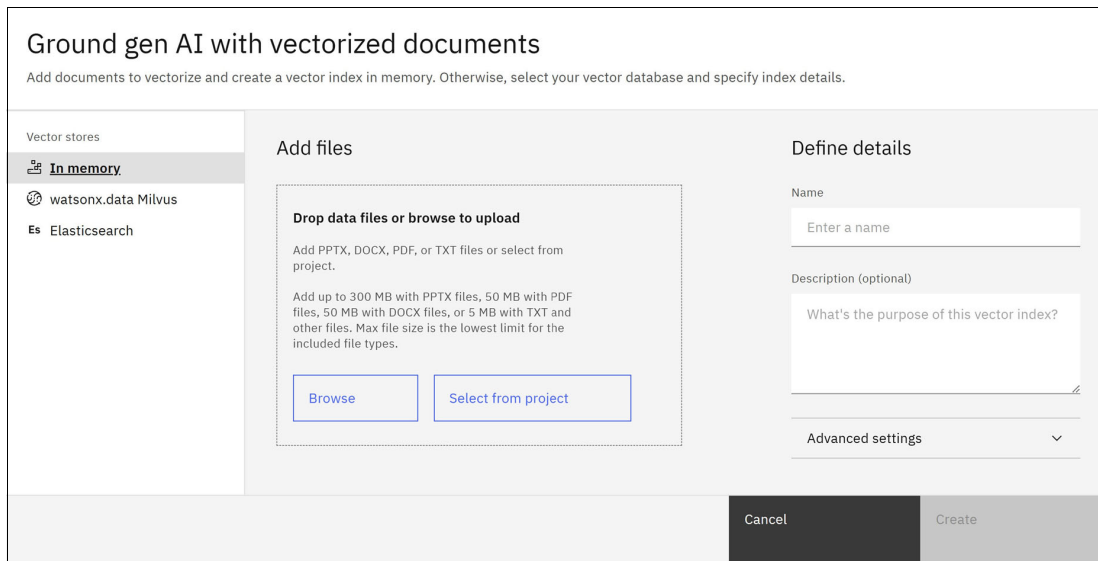


Figure 5-3 Use in-memory, IBM watsonx.data, or other data storage in Prompt Lab

The Prompt Lab UI is useful for exploring information or performing quick tests. It enables interactions with foundation models without requiring extensive code and provides an interface with guided options to support initial experimentation and further development.

5.1.2 Notebook for a programmatic workflow

For repeatable tasks, users typically migrate to writing code and running custom applications to automate AI workflows. watsonx.ai includes integrated Jupyter Notebook support for developing and automating use cases, as shown in Figure 5-4. Notebooks store code; can be ran in blocks; and can be modified and rerun as needed. They provide an intuitive interface for automation and help simplify workflow development before moving to full application deployment.

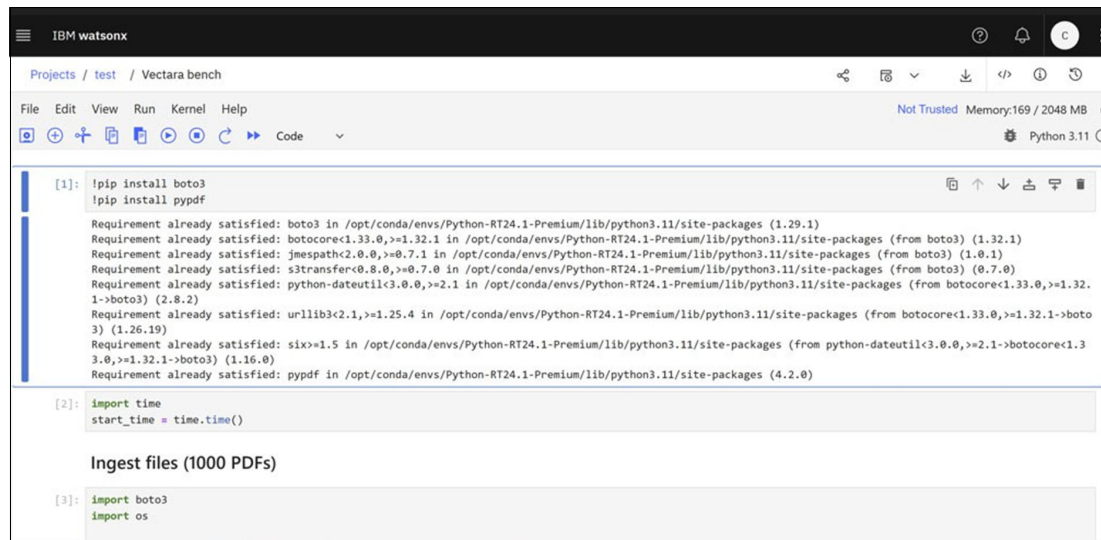


Figure 5-4 Jupyter Notebook UI example

The code listings in the following sections can be ran directly in such a notebook.

5.2 Use case: Robust entity extraction and summarization with Docling

Docling, which is an open-source project of the Linux Foundation, is a document-processing tool that addresses a common requirement in generative AI applications: extracting and structuring information from diverse document formats. By parsing formats such as PDFs, Office documents, audio files, and images, Docling supports capabilities including page-layout analysis, table-structure recognition, and optical character recognition (OCR). These functions are useful for context-engineering scenarios in which AI agents or LLMs require information that is stored in various document formats.

Docling provides integrations with AI frameworks such as LangChain, LlamaIndex, and Crew AI. Combined with its unified Docling Document format and multiple export options (Markdown, HTML, and JSON), it supports processing unstructured documents for use by AI agents. Docling can generate Markdown that is suitable for use with LLMs while preserving tables, reading order, and figures, which can improve downstream extraction quality and help reduce token usage.

By combining an S3-compatible storage layer that supports high-throughput parallel downloads with Docling for structure-aware PDF-to-Markdown conversion, organizations can build a predictable and high-performance watsonx.ai inference pipeline. This section describes the architecture, an implementation pattern, and a sample implementation for on-premises environments.

5.2.1 High-level workflow overview

The high-level workflow that is implemented by the architecture, as shown in Figure 5-5, follows these steps:

1. The application receives a processing request.
2. A worker or service lists matching files in IBM Storage Scale by using the S3 API.
3. Each file is converted by Docling into Markdown. This conversion can be CPU- or GPU-bound and can be parallelized across multiple workers.
4. The Markdown output is passed to watsonx.ai by using a short, deterministic prompt template for tasks such as entity extraction or summarization.
5. The results (entities, summaries, and metadata) are written back to a results store, such as an IBM Storage Scale object store, and can optionally be added to a vector database for RAG use cases.

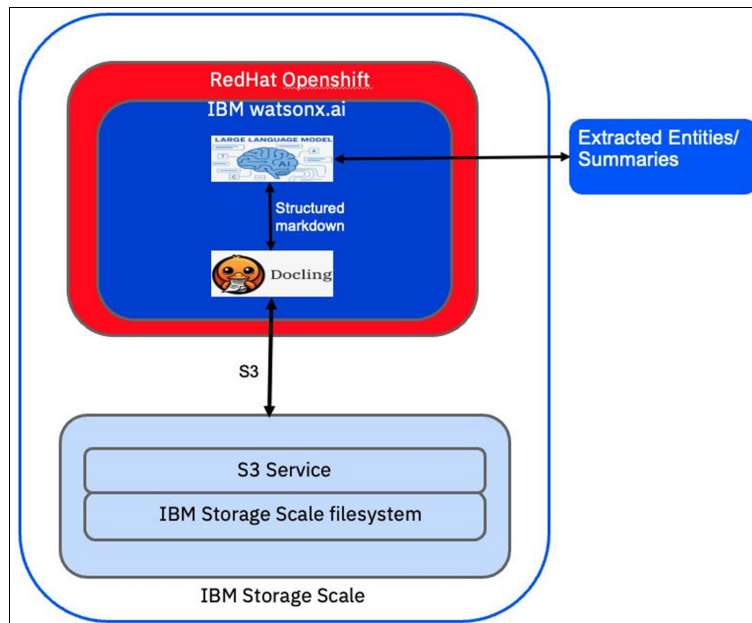



Figure 5-5 Content extraction overview

5.2.2 Financial summary example

The following example shows a compact Python workflow that demonstrates the core steps: fetching data from storage, processing it by using Docling, and then using an LLM for entity extraction and summarization.

Data preparation

Figure 5-6 shows tabular data along with other textual content. It does not have visible column borders, and similar variations in data layout are common. It is important to preserve the original structure before sending the content to the LLM because the loss of layout information can lead to misinterpretation and inaccurate outputs.



Key Financial Metrics

5

- Gross and pre-tax margin expansion, led by services
- Expense E/R reflects ongoing productivity and lower level of workforce actions, mitigated by lower IP income
- Net income and operating earnings per share reflect significant tax headwind
- Solid free cash flow performance supports investment and shareholder returns

P&L Highlights	1Q19	B/(W) Yr/Yr	P&L Ratios - Operating	1Q19	B/(W) Yr/Yr
Revenue	\$18.2	(1%)	Gross Profit Margin	44.7%	0.9 pts
Cloud & Cognitive Software	\$5.0	2%	Expense E/R	32.4%	2.2 pts
Global Business Services	\$4.1	4%	Pre-Tax Income Margin	12.3%	3.2 pts
Global Technology Services	\$6.9	(3%)	Tax Rate	10.0%	(40.6 pts)
Systems	\$1.3	(9%)	Net Income Margin	11.0%	(0.9 pts)
Pre-Tax Income - Operating	\$2.2	28%	Cash Highlights		LTM
Net Income - Operating	\$2.0	(12%)	Free Cash Flow (excl. GF Receivables)	\$1.7	\$12.2
Earnings Per Share - Operating	\$2.25	(8%)	Share Repurchase (Gross)	\$0.9	\$4.6
			Dividends	\$1.4	\$5.7
			Cash Balance @ March 31	\$18.1	

Revenue growth rates @CC, \$ in billions

Figure 5-6 Financial summary document example

The code in Example 5-1 demonstrates how unstructured documents can be converted into Markdown format in a few lines of code while preserving their original layout.

Example 5-1 Sample code to convert a PDF document into Markdown

```

from docling.document_converter import DocumentConverter

source = " IBM-Earnings-data.pdf "

converter = DocumentConverter()
result = converter.convert(source)

markdown_content= result.document.export_to_markdown()

# Print Markdown
print(markdown_content)

```

Figure 5-7 on page 43 shows the Docling output in Markdown format.

```
## Key Financial Metrics
```

- □ Gross and pre-tax margin expansion, led by services
- □ Expense E/R reflects ongoing productivity and lower level of workforce actions, mitigated by lower IP income
- □ Net income and operating earnings per share reflect significant tax headwind
- □ Solid free cash flow performance supports investment and shareholder returns

P&L Highlights	1Q19	B/(W) Yr/Yr
Revenue	\$18.2	(1%)
Cloud& Cognitive Software	\$5.0	2%
Global Business Services	\$4.1	4%
Global Technology Services	\$6.9	(3%)
Systems	\$1.3	(9%)
Pre-Tax Income - Operating	\$2.2	28%
Net Income - Operating	\$2.0	(12%)
Earnings Per Share - Operating	\$2.25	(8%)

P&L Ratios - Operating	1Q19	B/(W) Yr/Yr
Gross Profit Margin	44.7%	0.9 pts
Expense E/R	32.4%	2.2 pts
Pre-Tax Income Margin	12.3%	3.2 pts
Tax Rate	10.0%	(40.6 pts)
Net Income Margin	11.0%	(0.9 pts)
Cash Highlights		LTM
Free Cash Flow (excl. GF Receivables)	\$1.7	\$12.2
Share Repurchase (Gross)	\$0.9	\$4.6
Dividends	\$1.4	\$5.7
Cash Balance @March31	\$18.1	

Figure 5-7 Financial summary that is transformed to Markdown

LLM setup and prompt for extracting entities from the document

An LLM can use the processed document to extract structured entities by applying a prompt, such as in Example 5-2.

Example 5-2 Code to initialize the LLM

```
from ibm_watsonx_ai.metanames import GenTextParamsMetaNames as GenParams
from ibm_watsonx_ai.foundation_models.utils.enums import DecodingMethods
from langchain_ibm import WatsonxLLM

parameters = {
    GenParams.DECODING_METHOD: DecodingMethods.SAMPLE.value,
    GenParams.MAX_NEW_TOKENS: 2000,
    GenParams.MIN_NEW_TOKENS: 1,
    GenParams.TEMPERATURE: 0.5,
    GenParams.TOP_K: 50,
    GenParams.TOP_P: 1
}

llm = WatsonxLLM(
    model_id="ibm/granite-3-2-8b-instruct",
    url=credentials["url"],
    apikey=credentials["apikey"],
    project_id=project_id,
    params=parameters
)
```

Usage example

Example 5-3 shows an LLM prompt for extracting entities from the PDF to use the Docling output.

Example 5-3 LLM instruction example

```
template="""You are a skilled Financial Assistant responsible for analyzing equity
research reports and extract important entities.
```

```
You will be given an equity research report in markdown format. Extract the
following entities for quarter 1 of 2019 and present them in JSON format:
```

```
Revenue for Q1, 2019 in $ billions
- Gross profit margin for Q1, 2019
```

```
-----
OUTPUT FORMAT (STRICT)
```

```
-----
Return ONLY valid JSON text in the exact structure below:
```

```
---
{
  "Revenue" : "x.x",
  "Gross profit margin" : "x.x%"
}
---
```

Rules:

- The output must be valid JSON with no text outside JSON.
- All numbers must be strings (e.g., "7.5").

Markdown report:

```
{{markdown_data}}
```

JSON output: ""

```
prompt = PromptTemplate(
  template=template,
  input_variables=['markdown_data'],
  template_format="jinja2"
)
prompt_text = prompt.format(markdown_data=markdown_content)
model_output_text = None
gen = llm.generate([prompt_text], temperature=0)
print(gen.generations[0][0].text)
```

Output from the watsonx.ai LLM

Example 5-4 shows the information that the LLM extracted in the format that we requested.

Example 5-4 watsonx output returned

```
{
  "Revenue": "18.2",
  "Gross profit margin": "44.7%"
}
```

Further extensions

For more complex scenarios, the example in this section may be extended. For example, for PDFs containing images, you can use the multi-modal visual language model (VLM) pipeline from Docling, as shown in Example 5-5.

Example 5-5 Sample code to use an IBM Granite VLM model to convert content

```
from docling.datamodel.base_models import InputFormat
from docling.document_converter import DocumentConverter, PdfFormatOption
from docling.pipeline.vlm_pipeline import VlmPipeline
from docling.datamodel.pipeline_options import (
    VlmPipelineOptions,
)
from docling.datamodel import vlm_model_specs

pipeline_options = VlmPipelineOptions(
    vlm_options=vlm_model_specs.GRANITEDOCLING_TRANSFORMERS,
)

converter = DocumentConverter(
    format_options={
        InputFormat.PDF: PdfFormatOption(
            pipeline_cls=VlmPipeline,
            pipeline_options=pipeline_options,
        ),
    }
)

doc = converter.convert(source=pdf_path).document
markdown_content= doc.export_to_markdown()
```

Example 5-5 demonstrates how combining IBM Storage Scale with a Docling-based data pipeline enables watsonx.ai LLMs to generate structured outputs. In this pattern, each component performs a specific role and can scale horizontally: IBM Storage Scale supports storage throughput, Docling converts documents into Markdown suitable for LLM processing, and watsonx.ai performs extraction and summarization. Together, these components support predictable performance for bulk file-processing use cases.

5.3 Use case: Bulk data ingest and query evaluation with Open RAG Benchmark

This section demonstrates a RAG use case to provide LLMs with new information. For this purpose, we use the [Vectara Open RAG Benchmark](#). The benchmark contains a dataset of 1,000 scholarly articles in PDF format and more than 3,000 queries for RAG testing. Of the 1,000 documents, 400 contain answers to the queries, and the remaining 600 documents are noise and are irrelevant to the queries.

For this paper, we use the Open RAG Benchmark as an API rather than as a benchmark to illustrate the functional process and to demonstrate a bulk data-ingest workflow. We load the 1,000 documents into the RAG pipeline and store them in the knowledge base, which is the vector database.

5.3.1 Data ingest

The data-ingest process first retrieves all files. The files are then parsed and partitioned into chunks. For each chunk, corresponding embeddings, which serve as vector representations, are created. The chunks and their embeddings are stored in the vector database.

Pulling files

We use an S3 client to list all files from an S3 bucket on IBM Storage Scale, as shown in Example 5-6.

Example 5-6 Listing all files from an S3 bucket on IBM Storage Scale

```
s3_client = boto3.client('s3', endpoint_url='[URL]',
                        aws_access_key_id='xxxxxx',
                        aws_secret_access_key='yyyyyyy')

bucket="b-scale-wxai-data"
folder="open_ragbench/PDFs/"
response = s3_client.list_objects_v2(Bucket=bucket, Prefix=folder, Delimiter="/")
```

Iterating through the list, we download all PDFs and store the file names, as shown in Example 5-7.

Example 5-7 Iterating through the list

```
files = []
for fileKey in response['Contents']:
    file = fileKey['Key']
    if (file.endswith(".pdf")):
        target = os.path.basename(file)
        s3_client.download_file(bucket, file, target)
        files.append(target)
```

Parsing and chunking

We use the downloaded files and a standard library to parse the files and use a text splitter for chunking, as shown in Example 5-8.

Example 5-8 Parsing the files and using a text splitter for chunking

```
pages = []
for file_path in files:
    # process the file
    loader = PyPDFLoader(file_path)
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=512,
        chunk_overlap=20,
        length_function=len,
        is_separator_regex=False,
    )
    pages = loader.load_and_split(text_splitter)
    for page in pages:
        passages.append(page.page_content)
```

Creating embeddings

We iterate through the chunks and use an embedding model that is provided by watsonx.ai to create vector embeddings, as shown in Example 5-9. The embedding model runs on the CPU and does not require a dedicated GPU.

Example 5-9 Creating embeddings

```
from ibm_watsonx_ai.foundation_models.embeddings import
SentenceTransformerEmbeddings
model = SentenceTransformerEmbeddings("all-MiniLM-L6-v2")
embeddings = []
for passage in passages:
    embedding = model.embed_query(passage)
    embeddings.append(embedding)
```

Ingesting the data into the vector DB

To ingest the data into the vector database, we first establish a database connection. We use the Milvus client library for that purpose, as shown in Example 5-10.

Example 5-10 Ingesting the data into the vector DB

```
uri = "https://[host]:[port]" # Construct URI from host and port
user = 'xxxxxx'
password = 'yyyyyy'
client = MilvusClient(uri=uri, user=user, password=password,
                      secure=True,
                      db_name="default",
                      server_pem_path="milvus.pem",
                      # digital certificate file
                      server_name="[host]")
```

When the connection is created, we define the data schema for the collection that stores the data, as shown in Example 5-11.

Example 5-11 Defining the data schema for the collection that stores the data

```
collection_name = "vectara_bench"
primary_key = FieldSchema(
    name="id",
    dtype=DataType.INT64,
    is_primary=True,
)

vector = FieldSchema(
    name="vector",
    dtype=DataType.FLOAT_VECTOR,
    dim=384,
)

passage_text = FieldSchema(
    name="passage",
    dtype=DataType.VARCHAR,
    max_length=2048
)

schema = CollectionSchema(
```

```
        fields = [primary_key, vector, passage_text]
    )
```

When the schema is defined, we create the data collection and then iterate through the chunks to insert them in the database, as shown in Example 5-12.

Example 5-12 Creating the data collection and then iterating through the chunks

```
client.create_collection(
    collection_name=collection_name,
    schema=schema,
    using="default"
)
id = 0
for (passage, embed) in zip( passages, embeddings ):
    data=[{"id": id, "vector": embed, "passage": passage}]
    res = client.insert(
        collection_name=collection_name,
        data=data
    )
    id += 1
```

5.3.2 RAG query

To demonstrate the RAG-enhanced query, we show how to set up the model and load the data collection. We then select one of the Open RAG Benchmark queries as an example and present the RAG-enhanced answer.

Model and collection setup

To work with the models that are provided by IBM watsonx.ai, you must configure an authentication mechanism. In this example, we use the IBM Granite® model, which runs on a GPU. The authentication and model instantiation are shown in Example 5-13.

Example 5-13 Authentication and model instantiation

```
wslib = access_project_or_space()
def get_credentials():
    return {
        "url" : os.getenv('RUNTIME_ENV_APSX_URL'),
        "token" : wslib.auth.get_current_token(),
        "instance_id": "openshift"
    }

model_id = "ibm/granite-3-8b-instruct"
parameters = {
    "decoding_method": "greedy",
    "max_new_tokens": 350,
    "repetition_penalty": 1
}
project_id = os.getenv("PROJECT_ID")
space_id = os.getenv("SPACE_ID")

llm = Model(
    model_id = model_id,
    params = parameters,
```

```
credentials = get_credentials(),
project_id = project_id,
space_id = space_id
)
```

To operate on the data of the vector database, we load the respective collection, as shown in Example 5-14.

Example 5-14 Loading the collection

```
index_params = MilvusClient.prepare_index_params()
index_params.add_index(
    field_name="vector",
    metric_type="L2",
    index_type="IVF_FLAT",
    index_name="vector_index",
    params={ "nlist": 128 }
)

client.create_index(
    collection_name=collection_name,
    index_params=index_params
)

client.load_collection(collection_name=collection_name)
```

Query example

The Open RAG Benchmark contains many queries. Example 5-15 shows an example of one of these queries.

Example 5-15 Query example

```
with open(queries_file) as f:
    queries_json = json.load(f)

# example query
question = queries_json["9199173b-3ed1-4118-88cd-1713fc5fa8a7"]
print(question)
```

Example 5-16 shows the query output.

Example 5-16 Query output

```
{'query': 'How do changes in effective microbial death rate influence parameters like alpha and beta?', 'type': 'abstractive', 'source': 'text'}
```

To obtain the RAG-enhanced answer, we first create the embedding of the question and then use it to retrieve the relevant context, as shown in Example 5-17. The question and the retrieved context form the RAG prompt that is sent to the LLM to produce the result.

Example 5-17 Creating the embedding of the question and retrieving the relevant context

```
question_embedding = model.embed_query(question)

num_results = 5
search_params = {
    "metric_type": "L2",
    "params": {"nprobe": 5}
}
res = client.search(
    collection_name=collection_name,
    data=[question_embedding],
    anns_field="vector",
    search_params=search_params,
    limit=num_results,
    output_fields=["passage", "role"],
)

relevant_chunks = []
for i in range(num_results):
    text = res[0][i].get('entity').get('passage')
    relevant_chunks.append(text)
context = "\n\n".join(relevant_chunks)

rag_prompt = f"Please answer a question by using the following context. " + \
f"If the question is unanswerable, say \"unanswerable\"." + \
f"If the answer can be found in a paragraph, include the paragraph in the response." + \
f"\n\nContext:\n\"{context}\"" + \
f"\n\nQuestion: {question}"
rag_response = llm.generate_text(prompt=rag_prompt, guardrails=False)
print(f"Question: {question}")
print(rag_response)
```

Example 5-18 shows the RAG-enhanced answer.

Example 5-18 RAG-enhanced answer

Question: How do changes in effective microbial death rate influence parameters like alpha and beta?

Answer: Changes in effective microbial death rate, represented by the decay rate ratio (β), have a significant influence on parameters like alpha. As the noise level increases, both alpha and beta decrease, indicating system instability. However, for stable oscillation, a middle level of decay rate ratio (β) is required. This is demonstrated by both indicators increasing to a peak at $\beta \cdot 1.8 \pm 0.2$ and then decreasing with higher β . This pattern is consistent across various noise levels, as shown in Figure S5.

References:

S3.1 Numerical procedure

As the parameterization $x(?)$ is not available analytically, for each set of rate noise levels and decay rate ratios (Figs.2C, D). Both indicators consistently demonstrate a clear trend with changing parameters. First, as the noise level increases, both indicators decrease, suggesting that the system becomes unstable. Second, both indicators exhibit a pattern of increasing to a peak at $\beta \cdot 1.8 \pm 0.2$ and then decreasing with higher β . This demonstrates that stable oscillation requires a middle level of decay rate ratio. Similar results under distinct noise levels are shown in Fig. S5.

Example 5-18 on page 50 shows the question and the RAG-enhanced answer. It also contains references to support the result. The example here is highly domain-specific, as are typical RAG use cases.

5.4 Use case: RAG with a role-based content filter on IBM Storage Scale

This section extends the previous use case by introducing a multi-tenant approach. In a typical RAG pipeline, the core knowledge base contains much information, and not all content is relevant for all users. One option is to shard the database and separate it by information domains, but this approach can result in many databases, potential data duplication, and increased management complexity. Another approach, which is demonstrated in the example in this section, augments the database and the RAG pipeline with a role-based content filter. This section highlights the specific extensions to the implementation from the previous use case.

5.4.1 Data formats and data preparation

A key distinction in this approach is the data format. The data is annotated with the roles that are permitted to access it. The following sections show how the data is loaded into the system, how role-specific data encoding is applied, and the resulting vector-database schema.

Data input

First, the data is pulled normally with an S3 client, as shown in Example 5-19.

Example 5-19 S3 data call

```
s3_client.download_file(bucket, file, target)
```

Next, the data is annotated specifically with the corresponding roles that are allowed to access the data, where access by multiple roles is also allowed. Hence, we use an array to encode the list of roles and for this example use 'T' for a Technical role, 'H': HR role. The encoding can be done per ingest stream or with individual logic. Here, we use two PDF files, one with salary information of the company and the second an IBM Storage Scale System data sheet containing technical specifications. We annotate the files directly, then we parse and chunk them.

Next, the data is annotated with the roles that are permitted to access it, and multiple roles can be assigned. An array is used to encode the list of roles; in this example, T represents a technical role and H represents an HR role. The encoding can be performed per ingest stream or applied through individual logic.

In this example, two PDF files are used: one containing company salary information and one IBM Storage Scale System data sheet containing technical specifications. The files are annotated, then parsed and chunked, as shown in Example 5-20.

Example 5-20 File parsing code

```
files = {
    "Information-Technology,roles-and-salaries.pdf": ["H"],
    "ibm-ess-data-sheet.pdf": ["T"],
}
passages = []
for file_path in files:
    role = files[file_path]
    # process the file
    loader = PyPDFLoader(file_path)
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=512,
        chunk_overlap=20,
        length_function=len,
        is_separator_regex=False,
    )
    pages = loader.load_and_split(text_splitter)
    for page in pages:
        passages.append([page.page_content, role])
```

Data encoding

When the data and corresponding vector embeddings are inserted into the vector database, the role information must be preserved. This information is encoded by using a bitmask: 01b indicates access for the technical role; 10b indicates access for the HR role; and 11b indicates access for both roles. This approach, which is shown in Example 5-21, can be extended to support more roles.

Example 5-21 Embedding code

```
model = SentenceTransformerEmbeddings("all-MiniLM-L6-v2")
embeddings = []
for passage in passages:
    embedding = model.embed_query(passage[0])
    embedding_value=0
    if 'T' in passage[1]:
        embedding_value+=1
    if 'H' in passage[1]:
        embedding_value+=2
    embeddings.append([embedding, embedding_value])
```

Vector database collection definition

For the schema of the vector database collection, we extend the regular collection with a “role” field when we create the collection, as shown in Example 5-22.

Example 5-22 Adding a role field and creating the collection

```
collection_name = "rb_rag"
primary_key = FieldSchema(
    name="id",
    dtype=DataType.INT64,
```

```

        is_primary=True,
    )

    vector = FieldSchema(
        name="vector",
        dtype=DataType.FLOAT_VECTOR,
        dim=384,
    )

    passage_text = FieldSchema(
        name="passage",
        dtype=DataType.VARCHAR,
        max_length=2048
    )

    role = FieldSchema(
        name="role",
        dtype=DataType.INT64
    )

    schema = CollectionSchema(
        fields = [primary_key, vector, role, passage_text]
    )

    client.create_collection(
        collection_name=collection_name,
        schema=schema,
        using="default"
    )

```

5.4.2 Model setup

The model collection remains the same as the regular RAG pipeline, as shown in Example 5-23.

Example 5-23 Model setup code

```

model_id = "ibm/granite-3-8b-instruct"
parameters = {
    "decoding_method": "greedy",
    "max_new_tokens": 350,
    "repetition_penalty": 1
}

llm = Model(
    model_id = model_id,
    params = parameters,
    credentials = get_credentials(),
    project_id = project_id,
    space_id = space_id
)

```

5.4.3 RAG queries

To demonstrate the RAG responses, we provide two examples in the following sections.

Technical role

First, we use an example for the technical role. We want to identify the important features of the IBM Storage Scale 6000. We provide the value that is associated with the technical role and use it when searching for the relevant context in the vector database, as shown in Example 5-24.

Example 5-24 User role addition

```
user_role = 1 # T
question = "What are the features of ESS 6000?"
question_embedding = model.embed_query(question)

num_results = 5
search_params = {
    "metric_type": "L2",
    "params": {"nprobe": 5}
}
res = client.search(
    collection_name=collection_name,
    data=[question_embedding],
    anns_field="vector",
    search_params=search_params,
    filter=f"role % 2 == {user_role}",
    limit=num_results,
    output_fields=["passage", "role"],
)
```

The search filter helps ensure that only data that is relevant to the technical role is retrieved. Based on the context that is returned from the RAG query, we construct the RAG prompt. Example 5-25 shows the RAG creation.

Example 5-25 RAG prompt creation

```
relevant_chunks = []
for i in range(num_results):
    text = res[0][i].get('entity').get('passage')
    relevant_chunks.append(text)
context = "\n\n".join(relevant_chunks)

rag_prompt = f"Please answer a question by using the following context. " + \
f"If the question is unanswerable, say \"unanswerable\"." + \
f"If the answer can be found in a paragraph, include the paragraph in the response." + \
f"\n\nContext:\n\n{context}\n\n" + \
f"\n\nQuestion: {question}"

print(rag_prompt)
```

Example 5-26 on page 55 shows the output of the RAG prompt.

Example 5-26 RAG prompt output

Please answer a question by using the following context. If the question is unanswerable, say "unanswerable". If the answer can be found in a paragraph, include the paragraph in the response.

Context:

"5Tiered StorageAnalytics workloads often require access to two different kinds of storage systems – some that use high-performance media for quickly reading and writing active data, and others that provide more cost-effective storage for less frequently accessed data. Storage Scale System 6000 is designed to allow organizations to dial in the exact balance of performance and capacity that are required by their workloads. Scale System 6000 can be configured with standard NVMe flash drives for maximum performance or

Storage Scale System 6000 is available in all-flash and hybrid configurations providing:–Up to 310 gigabytes per second (GBps) throughput with low latency;–Up to 13 millions IOPS using NVMeoF;–Up to 1.8PBe (effective capacity) in a standard 4U rack space. Storage Scale software is designed to enable the Storage Scale System 6000 to scale linearly so that throughput increases proportionally as more systems are added to a cluster. For sequential data and workloads requiring access to massive data datasets,

performance or with IBM FlashCore Modules when data density and compression are a higher priority. For workloads where storage capacity is important, Scale System 6000 supports up to nine expansion enclosures. The IBM Storage Scale Expansion Enclosure is an enterprise-class, fully redundant storage enclosure, containing up to 91 20-TB or 22-TB self-encrypting SAS hard disk drives (HDDs). Attaching eight expansion enclosures to the Storage Scale System 6000 expands the maximum storage capacity to 18 PB of HDDs

2Solution Brief – IBM Storage Scale & Storage Scale System Figure 1 – The IBM Storage Scale System 6000 can deliver up to 310 GBps throughput, up to 13 M IOPS, and up to 1.8PBe (effective capacity) in a 4U rack.

file system much more quickly than when checkpointing directly to object storage. With AFM, the checkpointed data can be asynchronously sent to object storage in a way that does not gate progress of the training job. Scale System 6000 is a certified storage solution for NVIDIA DGX SuperPOD. Storage Scale System 6000 supports the NVIDIA GPUdirect Storage protocol, which enables a direct data path between GPU memory and local or remote storage, such as NVMe or NVMe over Fabric (NVMe-oF). This GPUdirect"

Question: What are the features of ESS 6000?

With this prompt, we can query the LLM and obtain our result, which is shown in Example 5-27.

Example 5-27 LLM query call

```
rag_response = llm.generate_text(prompt=rag_prompt, guardrails=False)
print(f"Question: {question}")
print(rag_response)
```

Example 5-28 shows the LLM query output.

Example 5-28 LLM query output

Question: What are the features of ESS 6000?

Answer: The IBM Storage Scale System 6000, also known as ESS 6000, offers several features. It can be configured with standard NVMe flash drives for maximum performance or with IBM FlashCore Modules for higher data density and compression. The system provides up to 310 GBps throughput with low latency and up to 13 millions IOPS using NVMeoF. It has up to 1.8PBe (effective capacity) in a standard 4U rack space. The Storage Scale software enables the system to scale linearly, with throughput increasing proportionally as more systems are added to a cluster. For sequential data and workloads requiring access to massive data sets, ESS 6000 supports up to nine expansion enclosures. The IBM Storage Scale Expansion Enclosure contains up to 91 20-TB or 22-TB self-encrypting SAS HDDs. Attaching eight expansion enclosures to the Storage Scale System 6000 expands the maximum storage capacity to 18 PB of HDDs. ESS 6000 is a certified storage solution for NVIDIA DGX SuperPOD and supports the NVIDIA GPUDirect Storage protocol, enabling a direct data path between GPU memory and local or remote storage, such as NVMe or NVMe over Fabric (NVMe-oF).

References:

Solution Brief – IBM Storage Scale & Storage Scale System

Figure 1 – The IBM Storage Scale System 6000 can deliver up to 310 GBps throughput,

The question was answered within the relevant context.

HR role

In this example, we address a similar query from the HR role's perspective. This role can include access to sensitive information, such as company salary data. In this case, we want to identify the role of an IT System Administrator, as shown in Example 5-29.

Example 5-29 User role addition

```
user_role = 2 # H
question = "What is the role of an IT System Administrator?"
question_embedding = model.embed_query(question)

num_results = 3
search_params = {
    "metric_type": "L2",
    "params": {"nprobe": 5}
}
res = client.search(
    collection_name=collection_name,
    data=[question_embedding],
    anns_field="vector",
    search_params=search_params,
    filter=f"role >= {user_role}",
    limit=num_results,
    output_fields=["passage","role"],
)
```

When we search for relevant information, the filter again helps ensure that data is retrieved according to the assigned role. This filtered context is then used to create the RAG prompt that is shown in Example 5-30.

Example 5-30 RAG prompt creation

```
relevant_chunks = []
for i in range(num_results):
    text = res[0][i].get('entity').get('passage')
    text = text.replace("\n \n \n", "\n")
    relevant_chunks.append(text)
context = "\n\n".join(relevant_chunks)

rag_prompt = f"Please answer a question using only the following context. " + \
f"If the question is unanswerable, say \"unanswerable\"." + \
f"If the answer can be found in a paragraph, include the paragraph in the response." + \
f"\n\nContext:\n\"{context}\"" + \
f"\n\nQuestion: {question}"

print(rag_prompt)
```

Example 5-31 shows the RAG prompt output and context.

Example 5-31 RAG prompt output and context

Please answer a question using only the following context. If the question is unanswerable, say "unanswerable".If the answer can be found in a paragraph, include the paragraph in the response.

Context:

"IT System Administrator

- Responsibility for managing information systems in the company.
- Analyzing company needs regarding its information systems.
- Designing and implementing information systems.
- Providing technical support to users.
- Collecting and recording requirements for change.

2,869 - 6,326 EUR

IT Tester

- Responsibility for the testing of software applications with an emphasis on detecting errors in terms of functions, usability, and safety of software.

ICT-specialist

- Providing technical support and troubleshooting for hardware and software issues.
- Implementing and maintaining IT infrastructure including networks, servers, and cloud services.
- Helping the development and deployment of new IT solutions and systems.
- Monitoring and optimizing system performance and security.
- Collaborating with cross-functional teams to integrate technology into business processes.
- Conducting regular system audits and assessments to ensure compliance with industry standards.
- Training staff on new technologies and software applications.
- Developing and maintaining documentation for IT processes and procedures.

- Managing user accounts, permissions, and access controls.
- Staying updated with the latest technology trends and advancements to recommend improvements.
- Ensuring that data backup and disaster recovery processes are in place and effective."

Question: What is the role of an IT System Administrator?

Finally, we use the RAG prompt to obtain the answer to the question from the pipeline, as shown in Example 5-32.

Example 5-32 RAG response code

```
rag_response = llm.generate_text(prompt=rag_prompt, guardrails=False)
print(f"Question: {question}")
print(rag_response)
```

Example 5-33 shows the RAG prompt output.

Example 5-33 RAG prompt output

Question: What is the role of an IT System Administrator?

Answer: The role of an IT System Administrator involves managing information systems in the company, analyzing company needs regarding its information systems, designing and implementing information systems, providing technical support to users, and collecting and recording requirements for change. The salary range for this role is 2,869 - 6,326 EUR

In this example, the response includes both an answer to the question and the associated salary-range information. This type of output can be used in scenarios such as supporting internal job-interview processes.

Abbreviations and acronyms

AFM	Active File Management
AI	artificial intelligence
ALB	Application Load Balancer
CAS	Content-Aware Storage
CES	Cluster Export Services
CNSA	Container Native Storage Access
CSI	Container Storage Interface
DNS	domain name system
GPU	graphics processing unit
gRPC	Google Remote Procedure Call
HA	high availability
HDD	hard disk drive
IBM	International Business Machines Corporation
KV	Key/Value
LLM	large language model
NAS	network-attached storage
NLB	Network Load Balancer
PoC	proof of concept
PV	persistent volume
PVC	persistent volume claim
RAG	retrieval-augmented generation
RU	rack unit
SDS	software-defined storage
VLM	visual language model

Related publications

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

IBM Redbooks

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

- ▶ *Accelerating AI and Analytics with IBM watsonx.data and IBM Storage Scale*, REDP-5743
- ▶ *IBM Storage Scale System Introduction Guide*, REDP-5729

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

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ Docling

<https://www.docling.ai/>
<https://github.com/docling-project/docling>

- ▶ IBM Documentation for IBM watsonx.data

<https://cloud.ibm.com/docs/watsonxdata>

- ▶ IBM Fusion Content Aware Storage (CAS)

<https://www.ibm.com/docs/en/fusion-software/2.11.0?topic=services-content-aware-storage-cas>

- ▶ IBM Storage Scale

<https://www.ibm.com/products/storage-scale>

- ▶ IBM Storage Scale

<https://www.ibm.com/products/storage-scale>

- ▶ IBM watsonx.ai

<https://www.ibm.com/products/watsonx-ai>

- ▶ IBM watsonx.ai product documentation

<https://www.ibm.com/docs/en/watsonx/w-and-w/2.2.0>

- ▶ IBM watsonx.data

<https://www.ibm.com/products/watsonx-data>

- ▶ Product documentation for IBM Storage Scale
<https://www.ibm.com/docs/en/storage-scale>
- ▶ Product documentation for IBM Storage Scale System
<https://www.ibm.com/docs/en/storage-scale-system>

Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

Services from IBM Consulting

[ibm.com/services](https://www.ibm.com/services)

IBM Training

[ibm.com/training](https://www.ibm.com/training)



REDP-5765-00

ISBN 0738462543

Printed in U.S.A.

Get connected

