

Data Accelerator for AI and Analytics

Simon Lorenz

Gero Schmidt

TJ Harris

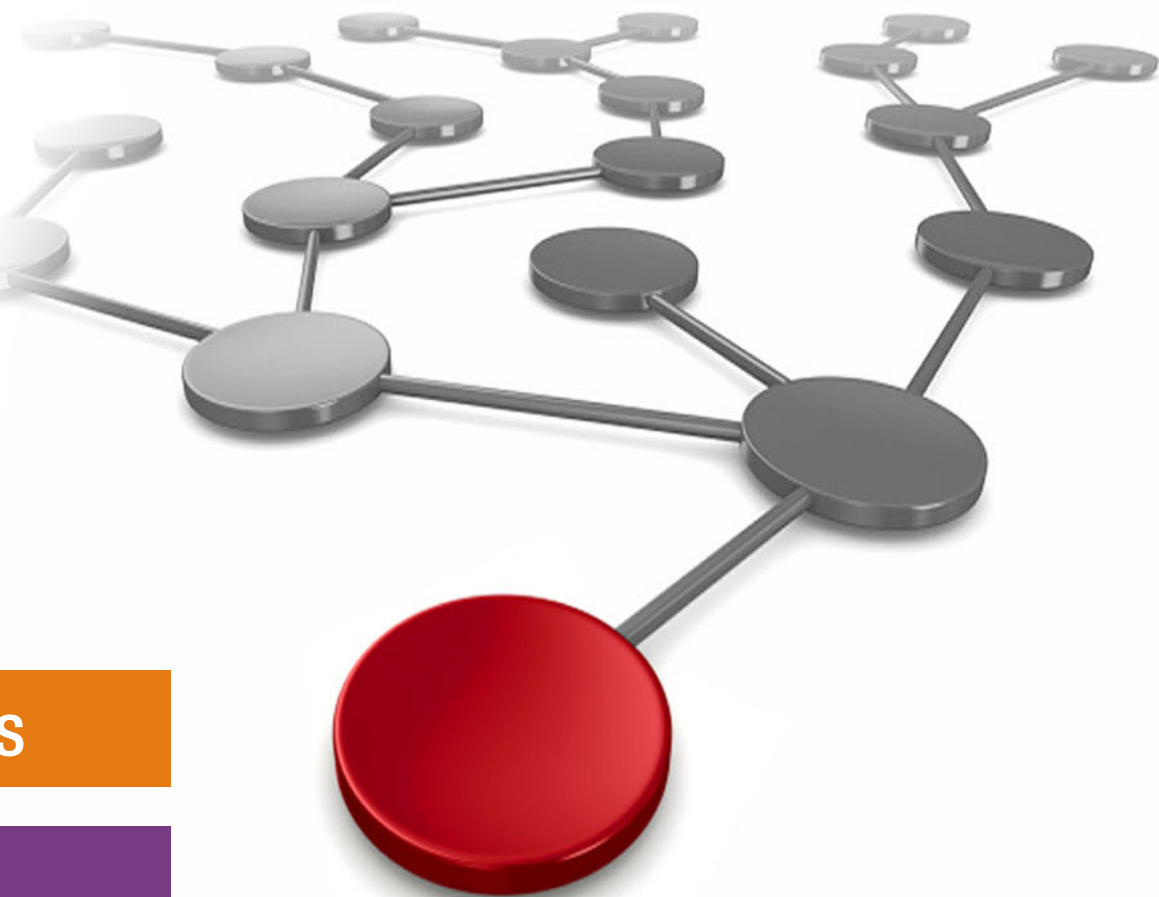
Mike Knieriemen

Nils Haustein

Abhishek Dave

Venkateswara Puvvada

Christof Westhues



 **Analytics**

Storage



IBM Redbooks

Data Accelerator for AI and Analytics

January 2021

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

First Edition (January 2021)

This edition applies to Version 5, Release 1, Modification 0 of IBM Spectrum Scale (product number 5737-F33).

© Copyright International Business Machines Corporation 2021. 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!	ix
Comments welcome	ix
Stay connected to IBM Redbooks	ix
Chapter 1. Data orchestration in enterprise data pipelines	1
1.1 Introduction	2
1.2 Overview	3
1.3 Sample use case: Building the correct training and validation data set	4
Chapter 2. Data Accelerator for AI and Analytics supporting data orchestration	7
2.1 Generic components	8
2.1.1 Data layer	8
2.1.2 High-performance storage with a smart data cache layer	9
2.1.3 Compute cluster layer	12
2.1.4 Data catalog layer	12
2.1.5 Interfaces between the layers	13
2.2 Proof of concept environment	14
2.2.1 Red Hat OpenShift V4.5.9 cluster	15
2.2.2 IBM Spectrum Scale V5.1.0 storage cluster	15
2.2.3 IBM ESS storage cluster	16
2.2.4 Capacity tier storage	16
2.2.5 IBM Spectrum Discover V2.0.2+ metadata catalog	16
2.2.6 IBM Spectrum LSF Workload Manager	16
2.2.7 Description of the Audi Autonomous Driving Dataset	17
Chapter 3. Data Accelerator for AI and Analytics use cases	19
3.1 Generic workflow	20
3.1.1 Provisioning phase	20
3.1.2 Analytic usage phase	21
3.2 Triggering an analytic job by using an integrated development environment	22
3.3 Workload manager starts an analytics job	24
3.4 New data ingest triggers an analytics job	24
3.5 The layer on top of workload triggers	25
Chapter 4. Planning for Data Accelerator for AI and Analytics	27
4.1 Security and data access rights considerations	28
4.2 Data layer	28
4.2.1 Network-attached storage (NSF) Filer	28
4.2.2 Cloud object storage	28
4.2.3 IBM Spectrum Archive Enterprise Edition Tape	30
4.3 High-performance storage with smart data cache layer	31
4.3.1 IBM ESS 3000 and IBM Spectrum Scale	31
4.4 Compute cluster layer	34
4.4.1 IBM Spectrum LSF	34

4.4.2 Compute cluster	34
4.5 Data catalog layer	37
4.5.1 IBM Spectrum Discover	37
Chapter 5. Deployment considerations for Data Accelerator for AI and Analytics . .	39
5.1 Data layer	40
5.1.1 Network-attached storage (NAS) Filer	40
5.1.2 IBM Cloud Object Storage	40
5.1.3 IBM Spectrum Archive Enterprise Edition Tape	41
5.2 High-performance storage with smart data cache layer	42
5.2.1 IBM ESS 3000 and IBM Spectrum Scale	42
5.3 Compute cluster layer	50
5.3.1 IBM Spectrum LSF	50
5.3.2 IBM Spectrum Scale storage cluster.	51
5.3.3 Compute cluster	56
5.4 Data catalog layer	65
5.4.1 IBM Spectrum Discover	65
5.5 The Data Accelerator for AI and Analytics interface glue code.	68
Appendix A. Code samples	71
Related publications	73
IBM Redbooks	73
Online resources	73
Help from IBM	74

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 <http://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.

Accesser®
IBM®
IBM Cloud®
IBM Research®

IBM Spectrum®
LSF®
POWER®
Redbooks®

Redbooks (logo) ®
Slicestor®

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.

LTO, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

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

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

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

Preface

This IBM® Redpaper publication focuses on data orchestration in enterprise data pipelines. It provides details about data orchestration and how to address typical challenges that customers face when dealing with large and ever-growing amounts of data for data analytics. While the amount of data increases steadily, artificial intelligence (AI) workloads must speed up to deliver insights and business value in a timely manner.

This paper provides a solution that addresses these needs: Data Accelerator for AI and Analytics (DAAA). A proof of concept (PoC) is described in detail.

This paper focuses on the functions that are provided by the Data Accelerator for AI and Analytics solution, which simplifies the daily work of data scientists and system administrators. This solution helps increase the efficiency of storage systems and data processing to obtain results faster while eliminating unnecessary data copies and associated data management.

Authors

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

Simon Lorenz is an IT Architect at IBM® Research and Development in Frankfurt, Germany. He joined IBM Germany in 1993 and has held various positions within IBM and IBM Research® and Development. During international assignments, he helped to improve fully automated chip factories in Asia and US. Simon joined the IBM Spectrum® Scale development team in 2014 and since then has worked on OpenStack Swift integration by designing and building System Health and Proactive Services solutions. Since March 2019, he is the worldwide leader of the IBM Spectrum Scale Big Data and Analytics team. His role includes designing and building solutions in the area of AI, such as DAAA.

Gero Schmidt is a Software Engineer at IBM Germany Research and Development GmbH in the IBM Spectrum Scale development group. He joined IBM in 2001 by working at the European Storage Competence Center (ESCC) in Mainz, Germany, where he provided technical presales support for a broad range of IBM storage products with a primary focus on enterprise storage solutions, system performance, and IBM POWER® systems. Gero participated in the product rollout of the IBM System Storage DS6000/DS8000 series, co-authored several IBM Redbooks® publications, and was a frequent speaker at IBM international conferences. In 2015, he joined the storage research group at the IBM Almaden Research Center in California, US, where he worked on IBM Spectrum Scale, compression of genomic data in next generation sequencing pipelines, and the development of a cloud-native backup solution for containerized applications in Kubernetes and Red Hat OpenShift. In 2018, he joined the Big Data & Analytics team in the IBM Spectrum Scale development group. He holds a degree in Physics (Dipl.-Phys.) from the Braunschweig University of Technology in Germany.

TJ Harris is a Senior Engineer and Master Inventor who is based in Tucson, Arizona. TJ joined IBM in 2001 after receiving a BS in computer engineering at the University of Arizona. He has spent his career working in the various parts of the IBM Storage portfolio. He has experience with tape, block, file, and object products, including both hardware and software. Much of his career was spent designing high availability (HA) and disaster recovery (DR) (HADR) solutions. His current position is the architect for IBM Spectrum Discover.

Mike Knieriemen is a Program Manager with IBM Cloud® in Chicago, Illinois, US. He has more than 20 years of experience in enterprise systems management, software sales, training, and cloud computing. He leads a team of technical cloud object storage (COS) consultants and architects who support the sale of IBM Cloud Object Storage solutions and training for IBM Cloud Object Storage clients. He holds a Bachelor of Science degree in Management from Purdue University, Indiana, US.

Nils Haustein is a Senior Technical Staff Member at IBM Systems group. He is responsible for designing and implementing backup, archiving, file, and object storage solutions in EMEA. He co-authored the book *Storage Networks Explained*. As a leading IBM Master Inventor, he has created more than 160 patents for IBM, and he is a respected mentor for the technical community worldwide.

Abhishek Dave is a Scrum Master and Test Lead with IBM India (Pune). He holds a Master of Computer Application degree. His current interests are hybrid COS, clusters, DR, and replication. Abhishek has about 17 years of product test experience with various enterprise-level file and block products like the Parallel Cluster File system, and Cluster Volume Manager, HA, file-level and block-level replication, testing, automation, storage area networks (SANs).

Venkateswara Puvvada is a Senior Software Engineer working on IBM General Parallel File System (IBM GPFS) development at IBM. He has 16 years of experience in computer software product architecture and development. He holds master's degree from Gulbaraga University, India. He is skilled in open source software, open standards, storage, file systems, COS, DR, migration technologies, and distributed caching.

Christof Westhues is a Client Technical Specialist at IBM Germany GmbH in the IBM Hybrid Cloud Solutions Group. He joined IBM in 2013 after working for Platform Computing since 2001, with a short side-step at Microsoft. During his entire career, he took care of customers in the high-performance computing (HPC) and technical computing arenas that use IBM Spectrum Load Sharing Facility (IBM LSF®) as the core scheduler and all corresponding add-ons. In this role, he worked as a technical presales resource and gave public and custom LSF training classes, consulting with and advising LSF customers.

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

Larry Coyne
IBM Redbooks, Tucson Center

Raul Saba, Scott Brewer, Juergen Reis, Harald Seipp, Aaron Palazzolo, Piyush Chaudhary, Wesley Jones, Pallavi Galgali, David Wohlford
IBM Systems

Marc Eshel, Vasily Tarasov, Frank Schmuck
IBM Research®

Adolf Hohl
NVIDIA

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

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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



Data orchestration in enterprise data pipelines

This paper focuses on data orchestration in enterprise data pipelines. It provides details about data orchestration and how to address typical challenges that customers face when dealing with large and ever-growing amounts of data for data analytics. While the amount of data increases steadily, AI workloads must speed up to deliver insights and business value in a timely manner.

This paper provides a solution that addresses these needs: Data Accelerator for AI and Analytics (DAAA). A proof of concept (PoC) is described in detail.

1.1 Introduction

AI workloads are growing exponentially. Customers across all industries are creating large data stores of petabytes (PB) to exabytes (EB) in size. Although most of the data is relatively cold, it still must be accessed periodically for trend analysis or revalidation of models. Figure 1-1 illustrates this scenario.

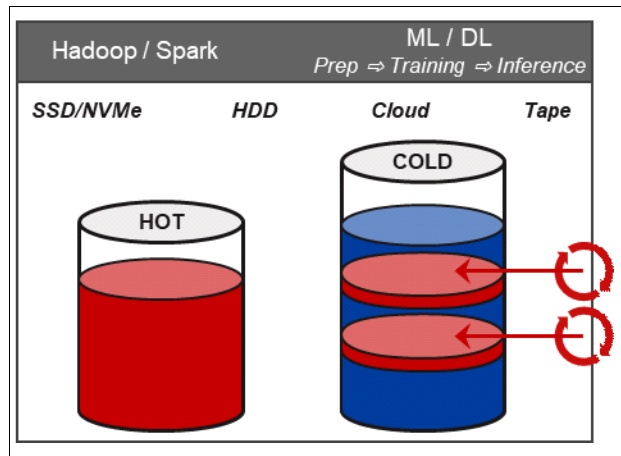


Figure 1-1 AI workloads must access relatively cold data for periodic trend analysis or revalidation of models

AI and analytic workloads require high-performance and low latency storage to keep expensive CPU, GPU, Tensor Processing Unit (TPU), and field-programmable gate array (FPGA) compute resources busy. Therefore, traditional data siloing is no longer appropriate.

There is a need for two types of storage tiers:

- ▶ A high-performance tier that provides maximum storage performance (\$/IOps and \$/GBps are key) with low latencies for random I/O operations (measured in input/output per second (IOPS)) and high bandwidths for total data throughputs (GBps).
- ▶ A capacity tier, that is, a data lake, for which minimizing the cost of storage \$/TB is key. It needs high durability, availability, and reliability guarantees, and geo-distribution capabilities.

With new faster storage technologies like NVMe and workloads running in the hybrid cloud, tiering functions alone are not enough. The high performance of today's data processing accelerators requires that data is closer to the accelerators faster and at the correct time.

Capacity tiers can be built from multiple different storage architectures and tiers, for example, hard disk drives (HDDs), network-attached storage (NAS), object storage, and tape. For a high-performance tier, it is essential to support as many storage architectures as possible to provide fast access to the requested data from the slower capacity tier to the analytic workloads in a single namespace.

Furthermore, today's workloads are more containerized, which raises the need to deeply integrate with schedulers and orchestrators.

As more components interact, management becomes increasingly complex. Where is the data? In which format is it available? A data scientist should not have to know about how and where to find and prefetch the data that needed, or what is needed to access it.

DAAA helps to address these needs.

1.2 Overview

DAAA is a solution that supports data orchestration. It helps to abstract data access across multiple storage systems and presents the data in a global namespace. It helps to support AI training at a large scale with the correct data and the availability of this data at the correct time.

How might an enterprise data orchestrated environment look? Figure 1-2 provides a high-level overview with layers.

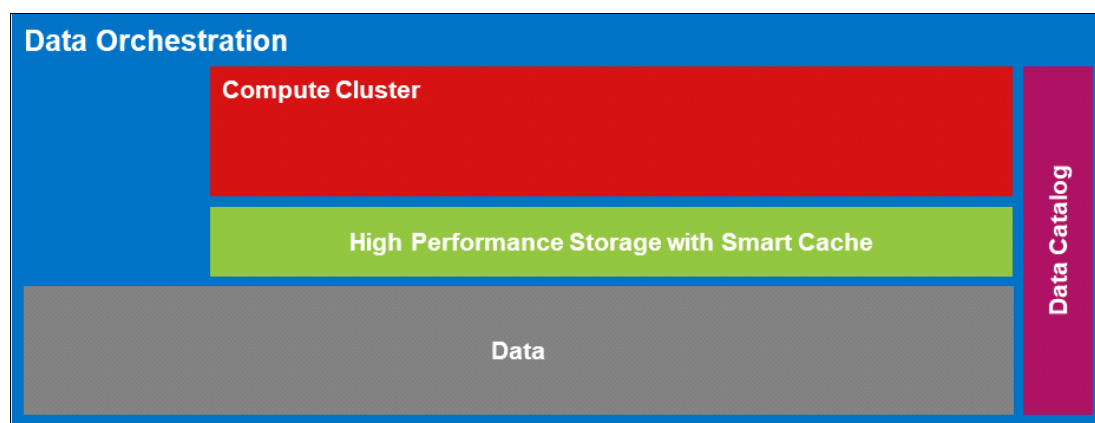


Figure 1-2 High-level example for an enterprise data orchestrated environment

It is all about data, which is the bottom layer.

In the end, you want to gain insights from the data, so it must be fed to the processing stage, which is represented by the compute cluster.

Customers across all industries are creating large PB to EB data stores, such as object stores. Today's computing resources are faster, so data access also must become faster.

Fast storage (for example, NVMe) is available, but it is too expensive for the underlying storage class for the capacities of today's data lakes. When doing data orchestration correctly, it is not needed at the capacity tier. A high performing smart storage tier with a smart data cache feeding the compute resources with the needed data is sufficient.

To find the correct data and get an idea about how long it takes to access it, a data catalog is needed.

A tight integration of such components can provide the required functions that simplify the daily work of data scientists and system administrators and help increase the efficiency of storage systems and data processing to obtain results faster while eliminating unnecessary data copies and associated data management.

1.3 Sample use case: Building the correct training and validation data set

In AI and deep learning (DL), building the correct training and validation data sets is one of the most challenging engineering tasks.

This section describes an automotive sample use case for a deep neural network (DNN) workload. For another PoC for this topic, see the real-world automotive industry training workload on a public data set that is documented in *Deployment and Usage Guide for Running AI Workloads on Red Hat OpenShift and NVIDIA DGX Systems with IBM Spectrum Scale*, [REDP-5610](#).

The validation of a trained neural network in various relevant situations gives you confidence about its performance and valuable insights into its weaknesses.

A rich set of metadata for the data set is crucial to build the necessary data sets and understand the neural network's strength and weaknesses.

The DNN that we used for our test performed well on large objects, but did not do well on smaller ones where the object occurrence was balanced. This situation also happened with the randomly chosen subset of our training data set.

We evaluated a training after 20,000 steps by using a confusion matrix, as shown in Figure 1-3.

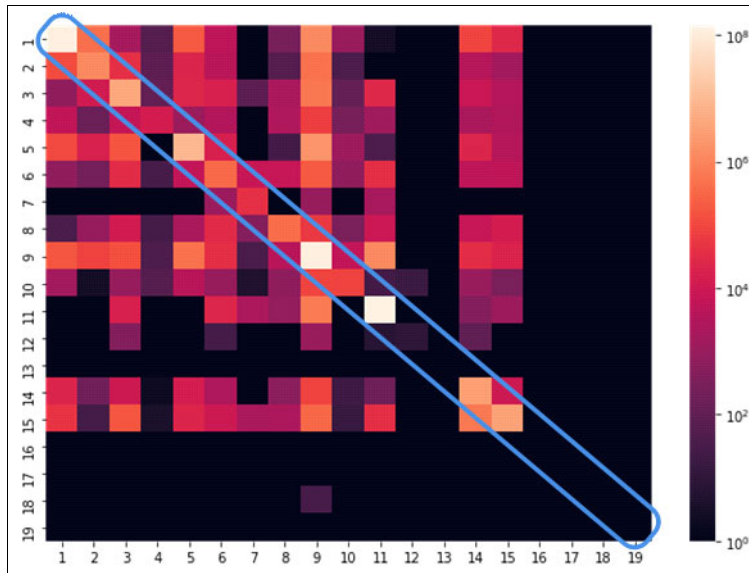


Figure 1-3 Confusion matrix presenting the training evaluation

For a perfect predictor, the diagram would have only a line from upper left to the lower right, which reads that the network correctly classified all pixels of a certain class.

There are two observations from this test:

- ▶ There are classes that can be detected well, such as the bright ones on the line: class 1 normal street, class 9 Sky, and class 11 Nature Object. When looking deeper at the metadata, you see that the frequency of those pixels occurring correlate. In the example, the normal street appears as 22%. In contrast, other classes such as “Road Blocks” appear with only 0.85%, and the frequency of object occurrence is not sufficient.
- ▶ It is obvious that the data that is used for the evaluation has no representatives in the example for bicycle class (19). Thus, the validation data set must be extended by adding frames that show bicycles.



Data Accelerator for AI and Analytics supporting data orchestration

Data Accelerator for AI and Analytics (DAAA) is a solution that addresses the needs of a data orchestrated environment.

It helps data scientists run their machine learning (ML) and deep learning (DL) (ML/DL) jobs without any knowledge about the underlying storage infrastructure, and find and prefetch the data that is needed.

DAAA can accelerate output from artificial intelligence (AI) and analytics applications in an enterprise data pipeline. It can bring the correct data at the correct time closer to AI and analytics applications for faster model training, inference, or real-time analytics without having to create another persistent copy of the data from the data lake.

DAAA can abstract warm, cold, and cold archive storage systems for the data scientist, and provides them with a data catalog to find and access the correct data for their analytics.

2.1 Generic components

With DAAA, you can use different components for each layer, as described in 1.2, “Overview” on page 3.

To demonstrate DAAA, we provide a solution with a sample architecture. The following section explains the components in detail and provides suggestions for different components that might be used for each layer.

Figure 2-1 shows components that might be part of the DAAA solution.

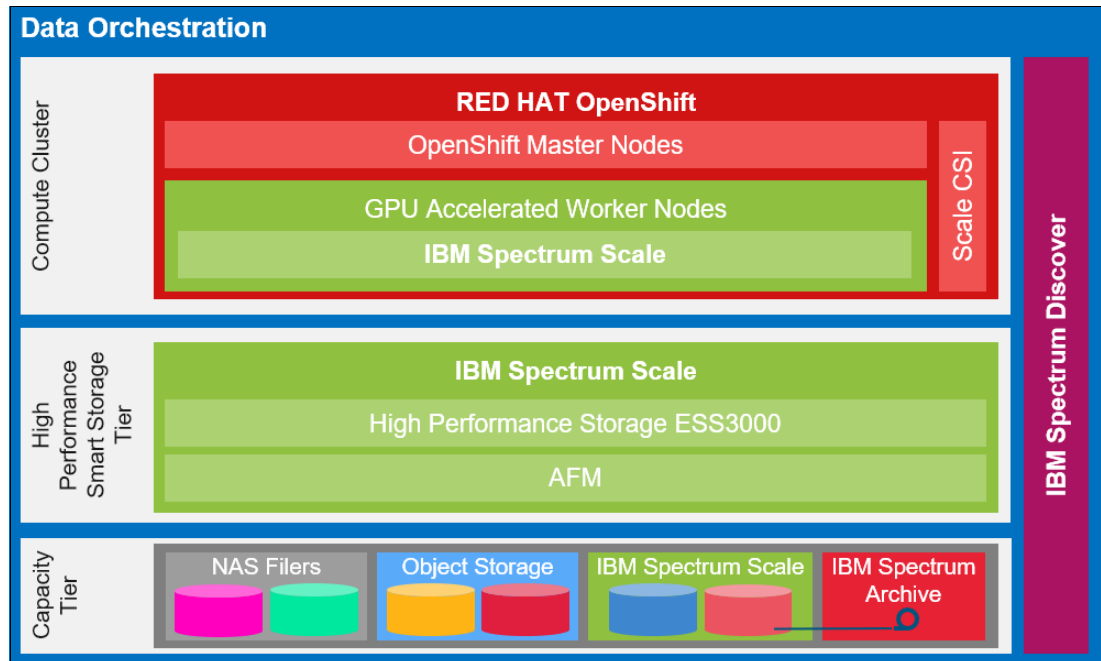


Figure 2-1 Example of an enterprise data orchestrated environment

2.1.1 Data layer

The data layer might also be named the *capacity tier* or *data lake*. It addresses storage for which cost (\$/TB); high durability, availability, reliability; and geo-distribution is key.

Figure 2-2 shows the capacity tier's (lowest layer in the diagram) different data storage systems, such as network-attached storage (NAS) Filers, object storage, or an IBM Spectrum Scale storage system to which an IBM Archive system also might be attached.

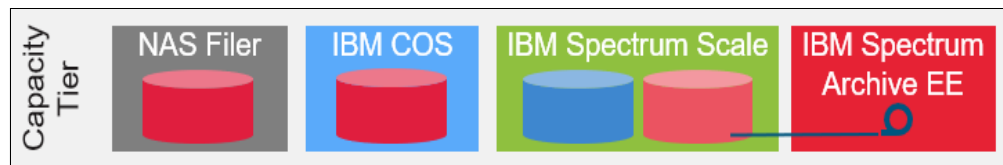


Figure 2-2 Capacity tier: Different data storage systems

Typically, different storage systems are used for different purposes. Some act as an ingest layer for new incoming data and must provide decent performance to receive the data from instruments. Another example is the growing needs of the automotive industry, where cars provide more data, either video, picture, and sensor data, which is used as a base to create autonomous driving solutions. Some of this data is live data after the car is sold and used by customers.

Costs correlate with storage performance and size. As data gets colder, object storage and tape solutions become more wanted. Additionally, the data location might be different: on-premises, in the cloud, or hybrid.

A smart data cache function must handle the different attached storage systems and storage locations.

In our sample proof of concept (PoC), shown in Figure 2-2 on page 8, we used an autonomous driving data set and stored parts in a Network File System (NFS) server that acts as a warm storage tier, an IBM Cloud Object Storage on-premises storage system that acts a cold storage tier, and IBM Spectrum Archive Enterprise Edition, which acts as a cold archive storage tier.

2.1.2 High-performance storage with a smart data cache layer

This layer can be described as the high-performance smart storage tier. It addresses storage that is maximized for performance, where \$/IOPps and \$/GBps, low latency random I/O, and high sequential bandwidth are key.

This layer acts as a cache that can provide the data fast to the AI and analytic workloads. The ownership of the data stays within the capacity tier storage systems.

To ensure data integrity, updates that are done on either layer must be done with consistency. If data is updated in the capacity tier, the data in the cache also must receive the update automatically, and vice versa. For the typical use cases that are addressed with the DAAA solution, data read workloads are in the majority.

For the high-performance storage and cache acting layer in our PoC, we use an IBM ESS 3000 storage system that combines IBM Spectrum Scale file management software with NVMe flash storage for the ultimate in scale-out performance and simplicity.

IBM Spectrum Scale provides the Active File Management (AFM) function to cache data from cost-optimized capacity tiers and data lakes like NAS Filers, IBM Cloud Object Storage, or other IBM Spectrum Scale clusters. It acts as a cache that fetches only the required data either on demand or on a schedule from the capacity tiers. In addition to a read to cache function, AFM also provides a write to cache function and ensures that any written data (such as analytic results) is flushed to the capacity tier.

In most cases with proper networking, you gain administration and resource sizing advantages when you separate the compute cluster from the storage cluster. In our use case, we separate the compute cluster from the storage cluster. The data that is cached in the IBM ESS 3000 storage cluster is remote-mounted by the IBM Spectrum Scale cluster that is part of the compute cluster. With this configuration, we can separate the combination compute cluster and high-performance smart storage tier from the capacity tier, as shown in Figure 2-3.

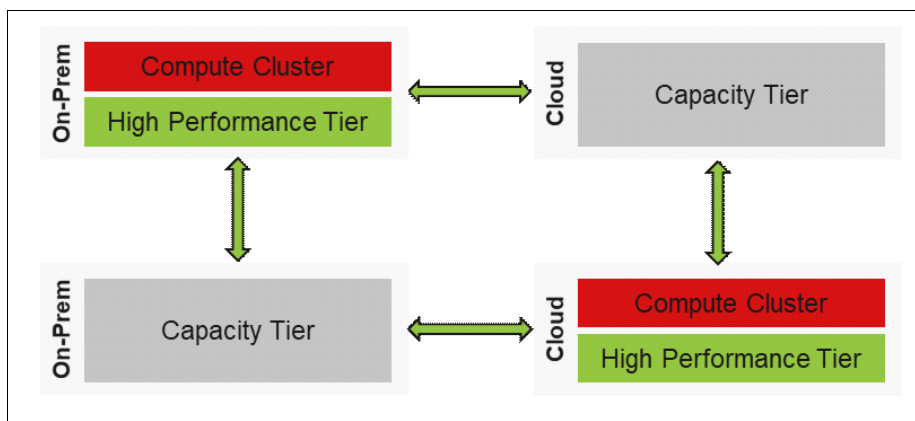


Figure 2-3 Hybrid cloud example: On-premises and cloud combinations

The compute cluster and high-performance smart storage tier may be on-premises or in the cloud, and the capacity tier is on-premises or in the cloud. All combinations are supported by the AFM function, so you can have use cases in which on-premises compute resources are exhausted in the short term and expanded into the cloud while the capacity tier stays on-premises.

Active File Management details

IBM Spectrum Scale AFM enables sharing of data across clusters, even if the networks are unreliable or have high latency. Figure 2-4 on page 11 shows how AFM acts as the smart data cache.

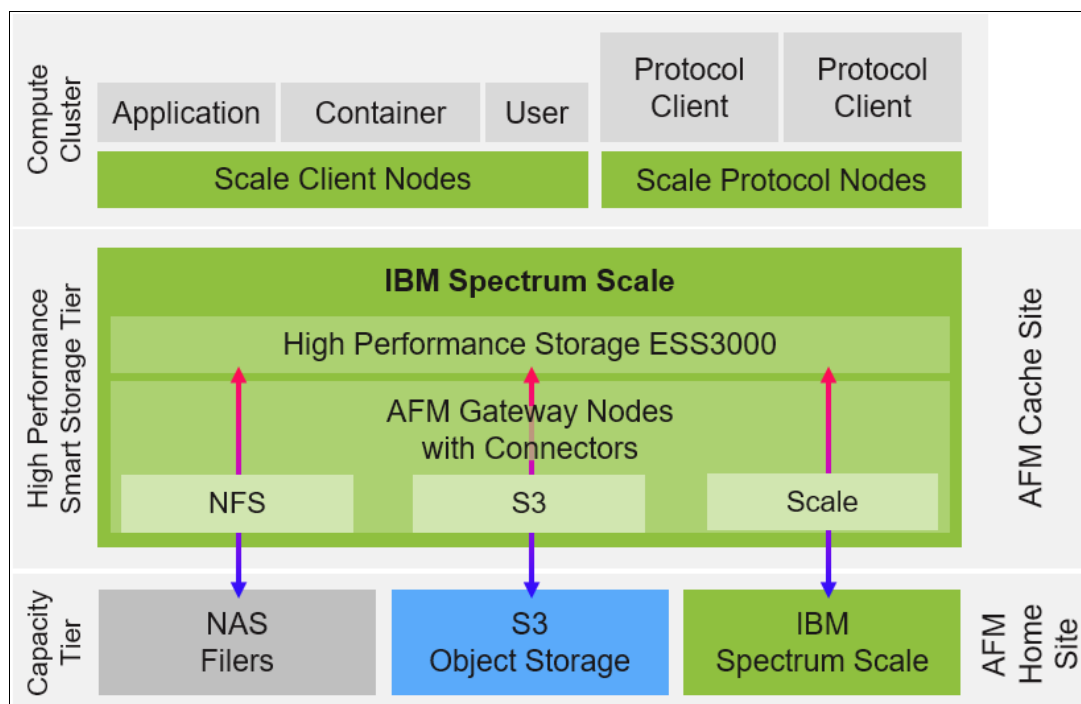


Figure 2-4 IBM Spectrum Scale Active File Management acting as a smart data cache

With AFM, you can create associations to NFS data sources, Amazon Simple Storage Service (Amazon S3) Object data sources, or further IBM Spectrum Scale clusters. With AFM, a single namespace view across sites around the world can be implemented, which makes the global namespace truly global.

AFM uses the terms *home site* and *cache site*. In DAAA, the Home site is the capacity tier, and the Cache site is the high-performance smart storage tier.

AFM constantly maintains an active relationship between the cache and home sites. Changes are managed by the IBM Spectrum Scale file set, which results in a modular and scalable architecture that can support billions of files and petabytes of data. Synchronization of the cache site works for both read (main DAAA use case) and writes.

AFM can prefetch data so that it is available when the analytic job starts. AFM caches data on the application request to ensure that any analytic job that is running in the compute cluster does not fail, for example, when a single file not available yet.

Prefetching data and caching an application request keeps data on both sites synchronized, which makes AFM the heart of the solution.

For more information about, IBM Spectrum Scale AFM, see [IBM Knowledge Center](#).

2.1.3 Compute cluster layer

This layer represents any kind of compute clusters running traditional or new generation applications, such as a Kubernetes or a Red Hat OpenShift cluster, as shown in Figure 2-5.

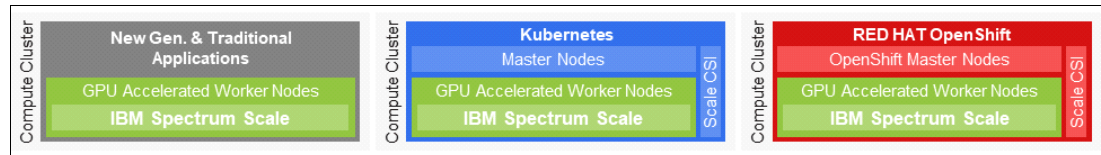


Figure 2-5 Example of different compute clusters with IBM Spectrum Scale and GPU-accelerated worker nodes

IBM Spectrum Scale nodes or clients that are part of the compute cluster remote mount the data that is stored in the connected IBM ESS storage cluster.

Traditional non-containerized applications can directly access the data in the IBM Spectrum Scale file system through the IBM Spectrum Scale client, but containerized applications running on container orchestration platforms such as Kubernetes or Red Hat OpenShift also require the IBM Spectrum Scale Container Storage Interface (CSI) driver to access data in IBM Spectrum Scale through *persistent volumes* (PVs).

For more information about IBM Spectrum Scale CSI, see [IBM Knowledge Center](#).

In our PoC, we used a Red Hat OpenShift V4.5.9 compute cluster with the IBM Spectrum Scale CSI Driver V2.0. We also used IBM Spectrum Load Sharing Facility (IBM LSF) as a workload manager that provided a customizable GUI window and acted as the data scientists interface to provide requests for the data that was needed.

2.1.4 Data catalog layer

This layer represents the key to the data. The data catalog layer holds the metadata of the data that is stored in the capacity tier, so you can easily search for the data that is needed.

Together with IBM Spectrum Scale AFM, the data catalog layer provides the correct data at the correct time for fast storage for analytic workloads.

DAAA is designed to allow integration with metadata management software such as IBM Spectrum Discover or other solutions like Starfish or pixit search to automate the selection of the correct data sets.

A data catalog must be able to catalog billions of files that are stored in different storage architectures. Search queries should be answered in a short time and must provide details about which storage system owns the data and where it is located.

Metadata in the best case should be pushed into the catalog, not pulled. If data scans are run, they must be lightweight but still fast enough to represent the details that are found in the data lake.

Other functions like finding duplicate data help to reduce the needed storage capacity.

2.1.5 Interfaces between the layers

The following sections provide more information about the interfaces that are used between the different layers.

Data to data catalog and high-performance storage

Data is scanned by the metadata search engine. If available, storage systems can also send data updates as an event to the metadata search engine, which reduces search load on the storage systems and provides real-time updates.

In our PoC, we used IBM Spectrum Discover as our metadata search engine, which can manage several connections to different storage systems. It can scan these systems by using policies and schedules. Scans are done by connecting to the storage system and reading the metadata that comes with the data. It is also possible to load metadata files that provide more details to the data. Furthermore, it provides functions to publish data about updated events into a Kafka queue to which IBM Spectrum Discover listens.

IBM Spectrum Scale AFM connects the data with the high-performance layer through the storage system by using interface protocols like NFS, Amazon S3, or Network Shared Disk (NSD) (IBM Spectrum Scale). AFM can sync the data in both directions (upwards and downwards) automatically. Data can be cached either on request or on-demand.

High-performance storage to data catalog and compute

In our PoC, we did not connect the metadata search engine to the high-performance storage because it was sufficient to show the concept without this connection. In a production environment, this connection helps to validate which data is already cached. Then, those details can be considered when you build the AFM prefetch or evict lists.

In our PoC, we connect an IBM Spectrum Scale Client cluster through a remote mount to the high-performance storage. IBM Spectrum Scale manages this connection. Depending on your environment and use case, the compute workloads might directly connect to the fast cache storage or if the workloads are in a separate compute cluster.

2.2 Proof of concept environment

Figure 2-6 shows the environment that we use for this PoC.

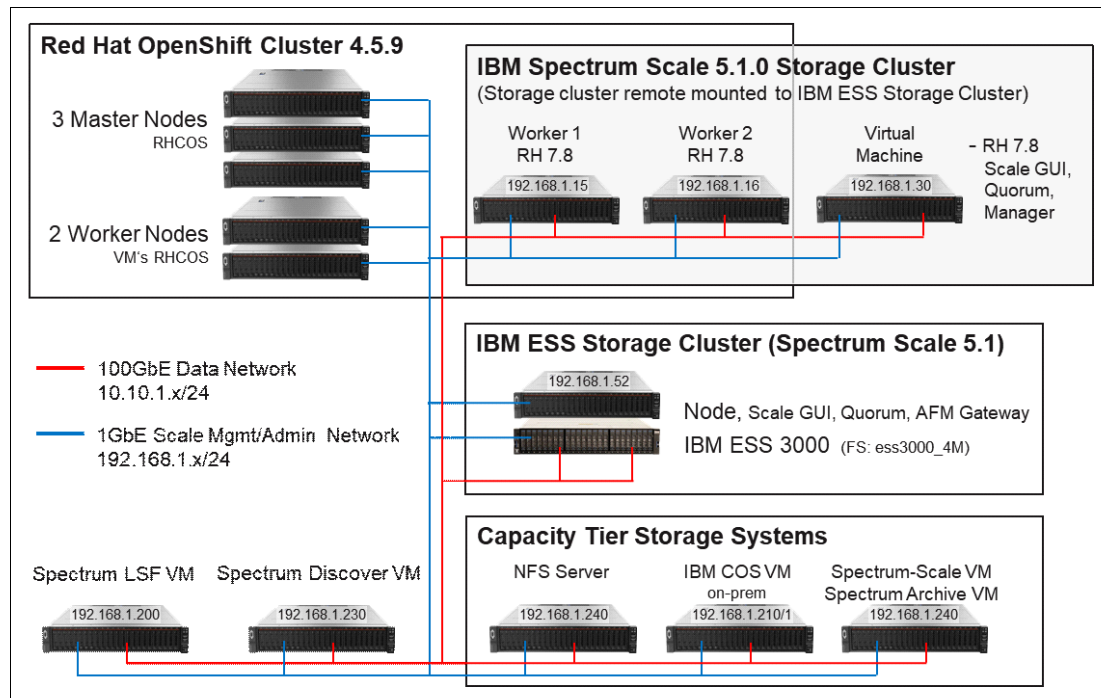


Figure 2-6 The environment that we use for this proof of concept

The solution that is described in this paper uses the following major components for the installation:

- ▶ Red Hat OpenShift V4.5.9 with access to Red Hat subscriptions and other external resources on the internet (for example, GitHub and container images registries)
- ▶ IBM Spectrum Scale Cluster V5.1.0 with IBM Spectrum Scale GUI and REST
- ▶ IBM ESS 3000 running IBM Spectrum Scale V5.1.0
- ▶ IBM Spectrum LSF V10.1.0 with IBM Spectrum LSF Application Center V10.2
- ▶ IBM Spectrum Discover V2.0.2 with extra functions added in Version 2.0.4, specifically the ImportTags embedded application
- ▶ NFSv3 server
- ▶ IBM Cloud Object Storage V3.14.7.56
- ▶ IBM Spectrum Scale V5.0.4.3 with IBM Spectrum Archive Enterprise Edition V1.3.0.7 connected to it
- ▶ Standard 100 Gbps Ethernet high-speed data network
- ▶ Standard 1 Gbps (or higher) Ethernet admin network for all components

Figure 2-7 on page 15 shows the software release levels that are used for Red Hat OpenShift and IBM Spectrum Scale, and the role of each node.

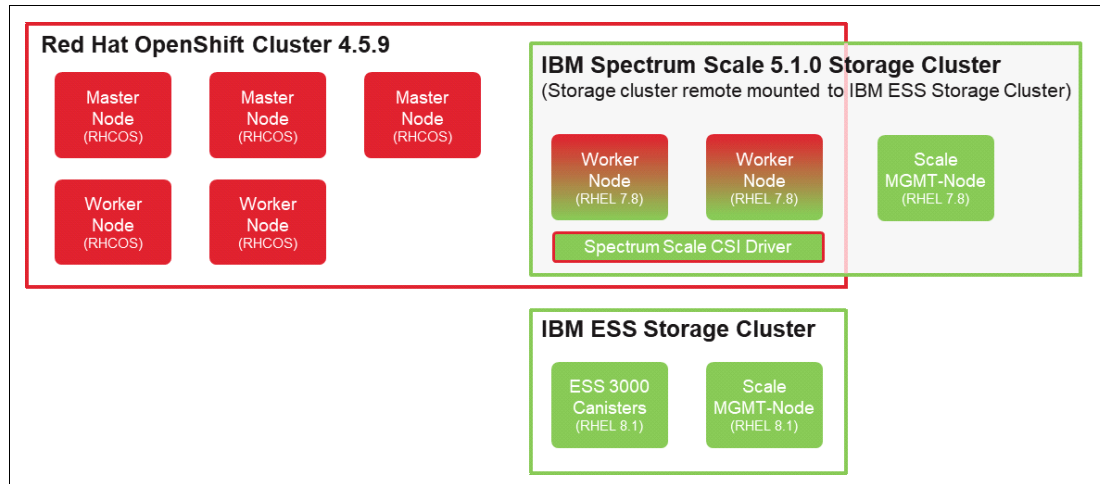


Figure 2-7 Red Hat OpenShift and IBM Spectrum Scale node roles and software releases

The following sections describe the clusters that were created for this PoC.

2.2.1 Red Hat OpenShift V4.5.9 cluster

Red Hat OpenShift is an open source container orchestration platform that is based on the Kubernetes container orchestrator. It is designed for enterprise app development and deployment. As an operating system, we deployed Red Hat Enterprise Linux CoreOS (RHCOS), and Red Hat Enterprise Linux (RHEL). For more information about RHCOS, see Appendix A, “Code samples” on page 71. RHCOS is the only supported operating system for Red Hat OpenShift Container Platform (Red Hat OCP) master node hosts. RHCOS and RHEL are both supported operating systems for Red Hat OCP on x86-based worker nodes. Because IBM Spectrum Scale is supported only on RHEL, we used RHEL V7.8 for these systems.

The compute cluster consists of the following items:

- ▶ Three master nodes running RHCOS on Lenovo SR650 servers
- ▶ Two worker nodes running RHCOS in a virtual machine (VM) on a Lenovo SR650 server (You must have a healthy Red Hat OCP cluster with a minimum of three master and two worker nodes running as a base before adding the GPU-accelerated workers.)
- ▶ Two GPU-accelerated worker nodes with NVIDIA GPU adapters

2.2.2 IBM Spectrum Scale V5.1.0 storage cluster

IBM Spectrum Scale is a high-performance and highly available clustered file system with associated management software that is available on various platforms. IBM Spectrum Scale can scale in several dimensions, including performance (bandwidth and input/output per second (IOPS)), capacity, and number of nodes or instances that can mount the file system.

The IBM Spectrum Scale storage cluster in our PoC consists of the following items:

- ▶ IBM Spectrum Scale clients running on every worker node, which are based on Red Hat V7.8
- ▶ IBM Spectrum Scale client running in a VM (providing GUI and REST interfaces, and quorum and management functions) that is based on Red Hat V7.8

- ▶ A remote-mounted IBM Spectrum Scale file system that is called `ess3000_4M` on an IBM ESS 3000 storage system that is configured with a 4 MiB blocksize (a good fit for the average image size of 3 - 4 MiB of the Audi Autonomous Driving Dataset (A2D2))

2.2.3 IBM ESS storage cluster

IBM ESS 3000 combines IBM Spectrum Scale file management software with NVMe flash storage for the ultimate scale-out performance and unmatched simplicity, and delivers 40 GBps of data throughput per 2U system.

The storage cluster consists of the following items:

- ▶ An IBM ESS 3000 storage system that has two canisters, each running IBM Spectrum Scale V5.1.0 on Red Hat V8.1. For more information about IBM Spectrum Scale V5.1 support on IBM ESS 3000, see the note at 4.3.1, “IBM ESS 3000 and IBM Spectrum Scale” on page 31.
- ▶ A2D2 downloaded and extracted into NFS, cloud object, and archive storage systems.
- ▶ A Lenovo SR650 server running IBM Spectrum Scale V5.1.0 on Red Hat V8.1 (providing GUI and REST, and quorum and management functions).

2.2.4 Capacity tier storage

For the capacity storage systems, we use the following items for this PoC:

- ▶ An NFS server that provides a single NFS share.
- ▶ An on-premises IBM Cloud Object Storage system with a single bucket.
- ▶ An IBM Spectrum Scale cluster with IBM Spectrum Archive Enterprise Edition connected to it.

For more information about IBM Spectrum Scale, IBM ESS 3000, IBM Spectrum LSF, IBM Spectrum Discover, IBM Spectrum Archive Enterprise Edition, IBM Cloud Object Storage, see “Related publications” on page 73.

2.2.5 IBM Spectrum Discover V2.0.2+ metadata catalog

For the metadata catalog, we use IBM Spectrum Discover. Connections to all configured capacity tier storage systems are created. For IBM Cloud Object Storage and IBM Spectrum Scale, we also configure the notification service that allows the storage systems to push events about metadata changes into the metadata catalog automatically.

In this solution, a pre-release version of IBM Spectrum Discover V2.0.4 is used. This upcoming release adds the ability to ingest pre-curated tags or labels for a data set.

2.2.6 IBM Spectrum LSF Workload Manager

For the workload manager to create jobs, we use IBM Spectrum LSF and the Application Center. They provide a customizable GUI and act as the data scientist's interface to make requests for the data that is needed.

2.2.7 Description of the Audi Autonomous Driving Dataset

Audi published the [A2D2](#), which can be used to support academic institutions and commercial startups working on autonomous driving research (for more information, see [A2D2](#)). The data set consists of recorded images and labels like bounding boxes, semantic segmentation, instance segmentation, and data that is extracted from the automotive bus. The sensor suite consists of six cameras and five LIDAR units, which provide 360-degree coverage. The recorded data is time synchronized and mutually registered. There are 41,277 frames with semantic segmentation and point cloud labels. Out of those frames, there are 12,497 frames that have 3D bounding box annotations for objects within the field of view of the front camera.

The semantic segmentation data set features 38 categories. Each pixel in an image has a label describing the type of object it represents, for example, pedestrian, car, or vegetation.

Figure 2-8 shows an example of a real picture compared to the segmentation picture.

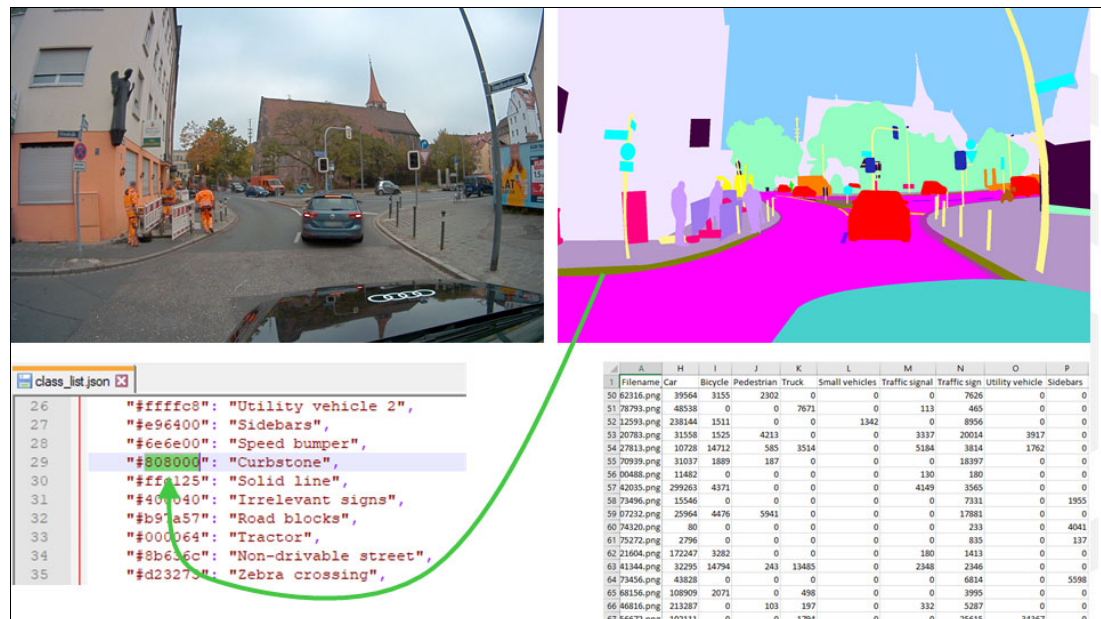


Figure 2-8 Real picture compared to the segmentation picture showing the color to label match

The data set was loaded into the different capacity tiers as follows:

- ▶ /camera_lidar_semantic/201811... and 201812... directories into the NFS storage
- ▶ /camera_lidar_semantic/201809... and 201810... directories into IBM Cloud Object Storage storage
- ▶ /camera_lidar_semantic/201808... directories into IBM Spectrum Scale / Archive storage



Data Accelerator for AI and Analytics use cases

This chapter describes provisioning and analytic usage workflows. The workflow use cases are based on a Red Hat OpenShift compute cluster environment. The workflows vary for different architectures. This chapter also describes three example use cases about starting the analytic usage workflow, which begins with a data scientist writing and creating the analytic job until new incoming data triggers the automated workflow.

3.1 Generic workflow

A workflow can be separated into phases that must be prepared by a system admin and phases that a user (in this use case, a data scientist) does.

Figure 3-1 shows an example.

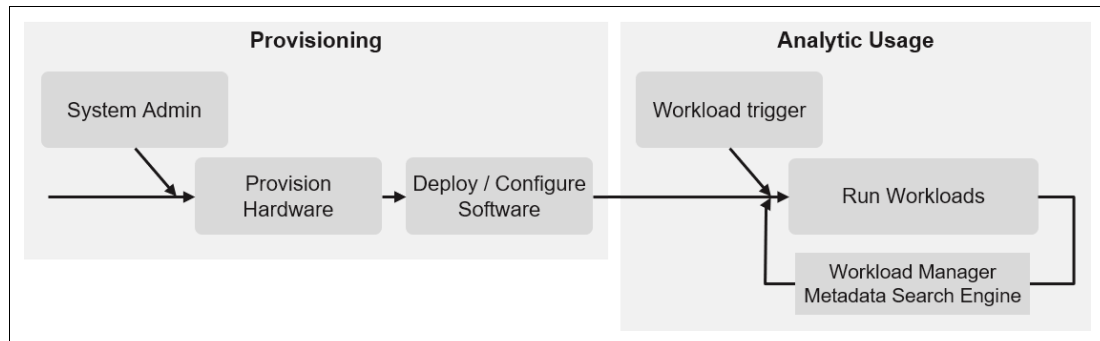


Figure 3-1 Provisioning and analytic usage phases

3.1.1 Provisioning phase

The provisioning phase sets up the computing power and software that is needed to use it.

Figure 3-2 provides an example matching our proof of concept (PoC) environment.

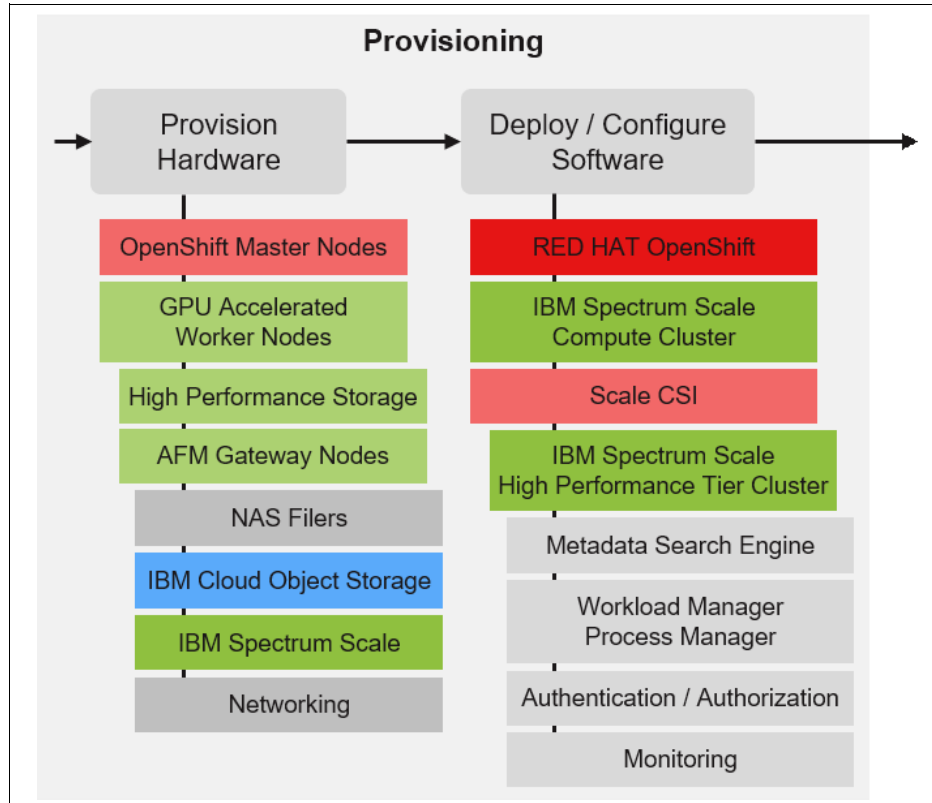


Figure 3-2 Provisioning phase

In this phase, your hardware and software planning depend on the size of your workloads. For more information, see Chapter 4, “Planning for Data Accelerator for AI and Analytics” on page 27.

3.1.2 Analytic usage phase

In the analytic usage phase, the provisioned hardware and deployed software are used for analytic workloads. For more information about possible triggers that create and start a job, see 3.2, “Triggering an analytic job by using an integrated development environment” on page 22, 3.3, “Workload manager starts an analytics job” on page 24, and 3.4, “New data ingest triggers an analytics job” on page 24.

Figure 3-3 provides an example workload flow that matches our PoC environment.

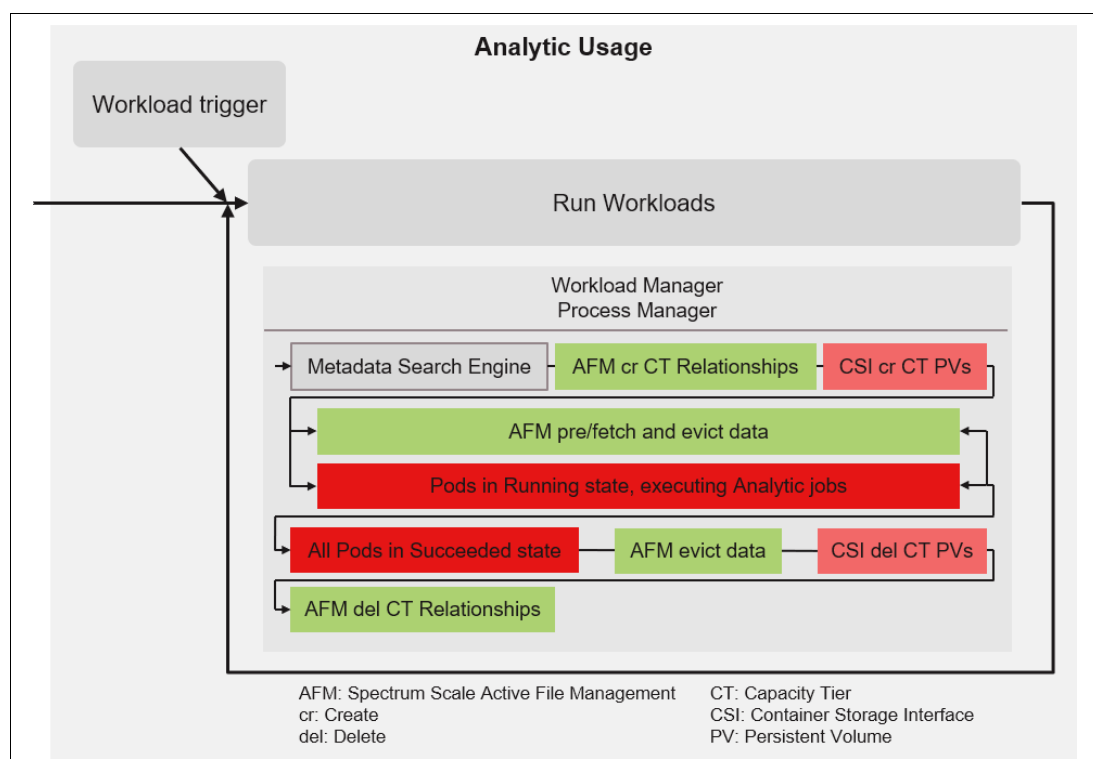


Figure 3-3 Analytic usage

Workflow triggers do not need to know about the underlying storage system where the data is, and how to find and prefetch the data that is needed.

Nevertheless, in our PoC environment, the following items must be provided to enable Data Accelerator for AI and Analytics (DAAA) to handle the machine learning (ML) and deep learning (DL) (ML/DL) jobs:

- ▶ Metadata tags that are used to address the data to be analyzed.
- ▶ User namespace that is used in the orchestrated environment.
- ▶ Name of the user's persistent volume claim (PVC) with the user's individual workspace and code that is used for running the analytics workload.
- ▶ Relative path to the code in the user's persistent volume (PV) (referenced through the PVC above) that is run for the analytics workload.

Extra computing preset templates can be provided by the system admin:

- ▶ Details about the resources that are needed (nodes, cores, mem, and other items)
- ▶ Capacity tier input data (network-attached storage (NAS) Filer, object storage, and the archive system)

A typical workload flow might look like the following example:

1. A workflow trigger provides the metadata search engine with a set of metadata tags and a set of paths to the data that holds the metadata tags. This data should be analyzed.
2. IBM Spectrum Scale Active File Management (AFM) relationships to the capacity storage are created if they do not exist. Users and their access rights to the data that is stored in the capacity tier must be considered.
3. The metadata search engine data that is needed is recalled from tape if it is located there. Requested data is prefetched by IBM Spectrum Scale AFM into high-performance storage.
4. The AFM prefetch process is monitored, and when the requested data is in place, PV manifest files, PVCs, and PVs are created.
5. Workloads running in OpenShift start Pods. AFM respects the file sequence of the prefetched data. If you know which data is needed first, analytic jobs can start when the portions of the needed data are available. Sequencing the prefetching list must be done too.
6. Pods are monitored. After all the involved Pods are in the “Succeeded” state, data is evicted from the shared high-performance storage. The overall job knowledge details might be needed to decide whether the data is also needed for a different job running in the environment. For more information, see 3.5, “The layer on top of workload triggers” on page 25.
7. After the job finishes, the PVCs and PVs are deleted.
8. AFM relationships that are controlled on the job request level that are no longer needed are evicted. In our PoC environment, we keep these relationships. Whether you do so depends on your environment’s size and the number of different attached storage systems and their data structure. For example, if your capacity tier consists of object storage with hundreds of buckets, it might make sense to create and remove relationships on demand.

3.2 Triggering an analytic job by using an integrated development environment

This section describes a workload that is manually triggered by a data scientist.

Data scientists typically use IDEs such as a Jupyter Notebook to create scripts that run the workload. It is possible to add code that interfaces with the IBM Spectrum Scale AFM function to prefetch and evict data to test the workload.

Figure 3-4 on page 23 shows an example.

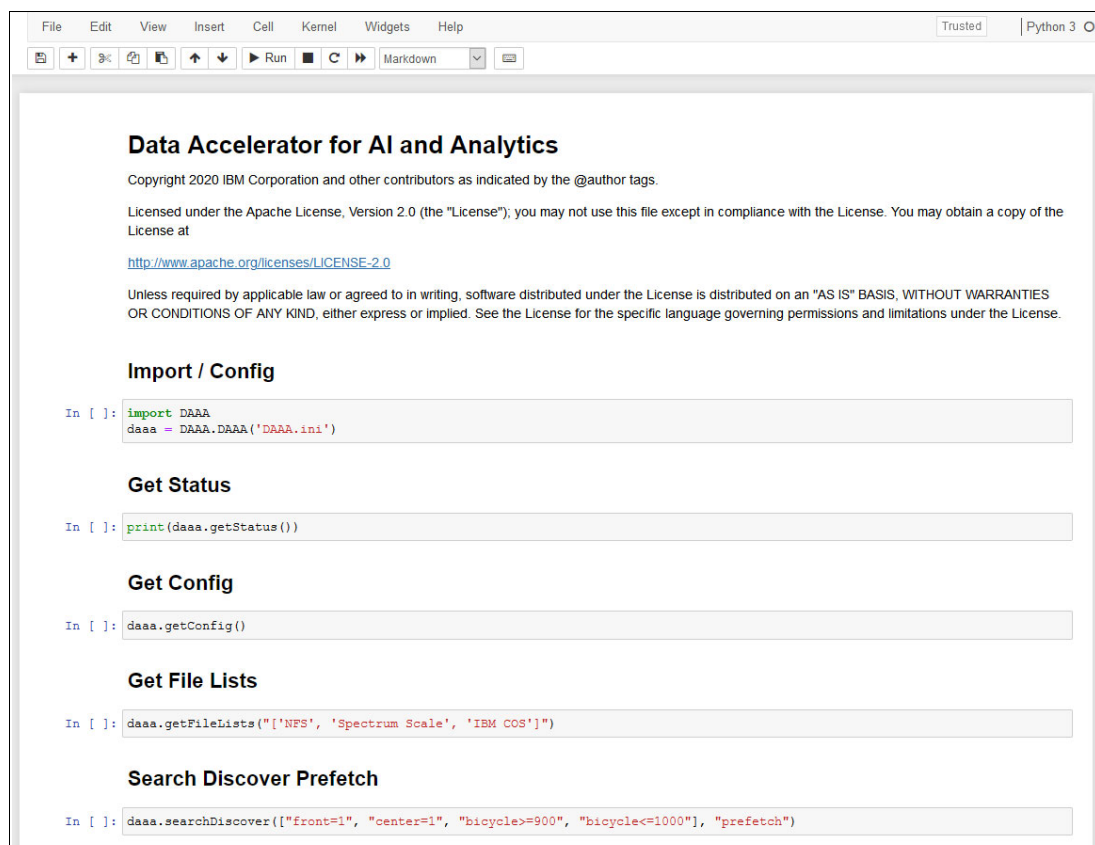


Figure 3-4 A DAAA Jupyter Notebook example

We create a sample that works with our PoC environment. A user may trigger every step (query discover, recall and cache data, and other steps) by clicking the **Play** button inside the Jupyter Notebook. With this action, a data scientist may interactively request specific data to be prefetched and made available in the high-performance tier, and the data is ready for analytics. For example scripts, see Appendix A, “Code samples” on page 71.

3.3 Workload manager starts an analytics job

This section describes a semi-automated workload. A user provides details to a job template that is submitted in the workload manager IBM Spectrum Load Sharing Facility (LSF).

Figure 3-5 provides a sample job submission window that is created in IBM Spectrum LSF and tailored for an autonomous vehicle workload.

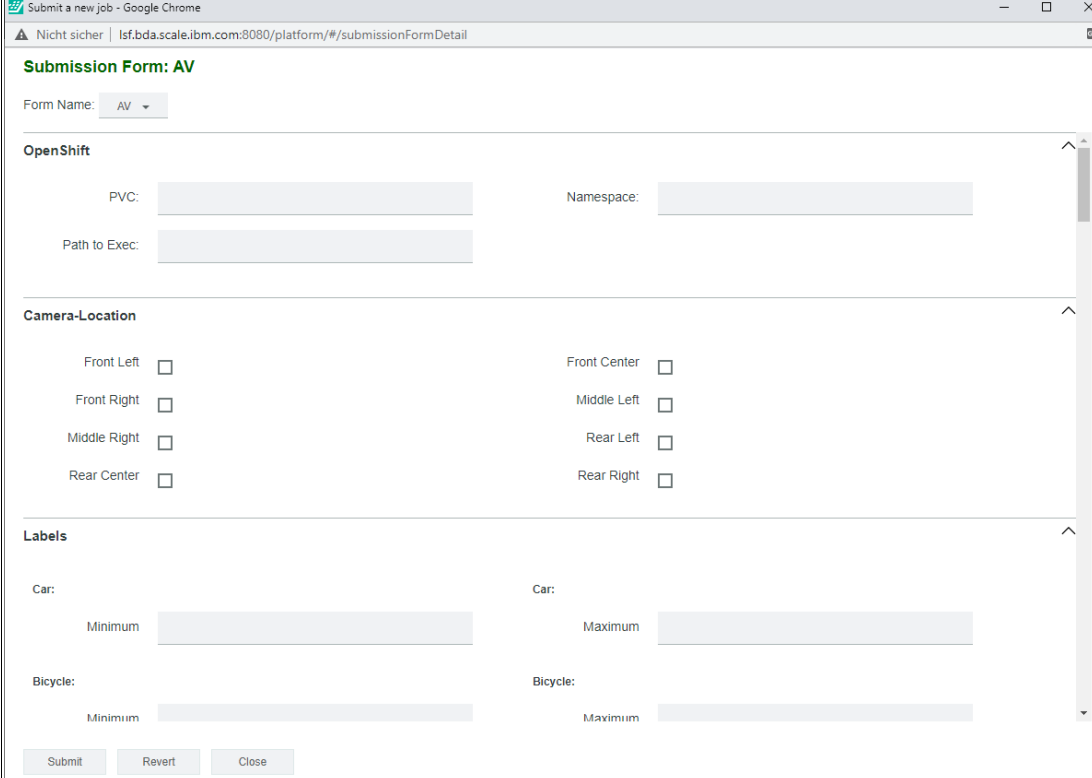
The screenshot shows a web browser window titled "Submit a new job - Google Chrome" with the URL "lsf.bda.scale.ibm.com:8080/platform/#/submissionFormDetail". The page is titled "Submission Form: AV". It contains several sections: "Form Name" with a dropdown menu set to "AV"; "OpenShift" section with input fields for "PVC:", "Namespace:", and "Path to Exec:"; "Camera-Location" section with checkboxes for "Front Left", "Front Right", "Middle Right", "Rear Center", "Front Center", "Middle Left", "Rear Left", and "Rear Right"; "Labels" section with input fields for "Car: Minimum", "Car: Maximum", "Bicycle: Minimum", and "Bicycle: Maximum". At the bottom are "Submit", "Revert", and "Close" buttons.

Figure 3-5 Autonomous vehicle workload in a tailored IBM Spectrum LSF job submission template

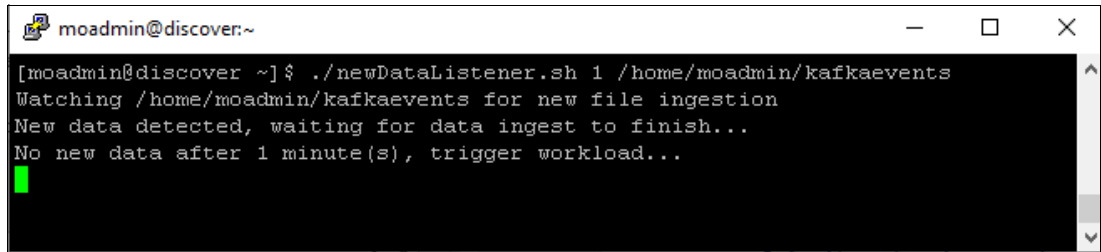
The template provides a data scientist with an easy method to enter details about the OpenShift environment that the job should run in, and the tags that describe the needed data.

After the data is entered and the user presses **Submit**, the entered data is collected and provided to a master job that is placed in IBM Spectrum LSF. The master job generates multiple subjobs that run the steps in 3.1.2, “Analytic usage phase” on page 21. For example scripts, see Appendix A, “Code samples” on page 71.

3.4 New data ingest triggers an analytics job

This section describes an automated workload trigger. Data ingest monitors might be watching the capacity tier for new incoming data. In our example, we used the IBM Spectrum Scale notification function. IBM Cloud Object Storage is configured to publish data update events into a Kafka queue that is connected to IBM Spectrum Discover.

Figure 3-6 on page 25 shows an example.

A terminal window titled 'moadmin@discover:~' with standard window controls. The terminal shows the execution of a script: [moadmin@discover ~]\$./newDataListener.sh 1 /home/moadmin/kafkaevents. The script outputs: 'Watching /home/moadmin/kafkaevents for new file ingestion', 'New data detected, waiting for data ingest to finish...', and 'No new data after 1 minute(s), trigger workload...'. A green cursor is visible on the line following the last output.

```
moadmin@discover:~  
[moadmin@discover ~]$ ./newDataListener.sh 1 /home/moadmin/kafkaevents  
Watching /home/moadmin/kafkaevents for new file ingestion  
New data detected, waiting for data ingest to finish...  
No new data after 1 minute(s), trigger workload...  
█
```

Figure 3-6 Monitor watching for new incoming data that triggers a master job after data ingest finishes

After no new events are received for a certain period, the monitor creates a preconfigured master job in IBM Spectrum LSF. The master job generates multiple subjobs that run the steps in 3.1.2, “Analytic usage phase” on page 21. For example scripts, see Appendix A, “Code samples” on page 71.

3.5 The layer on top of workload triggers

Data might be needed for a follow-on job or a finished job that is repeated with slightly different parameters.

Furthermore, multiple data scientists often use larger analytics environments and might need the same data in the high-performance cache.

If you are running such environments, the decision of when and which data should be prefetched and evicted should be made by an overall system that handles incoming and existing data scientist jobs. The system must have oversight for which jobs must work with which data, and for how long. With this oversight, the system can plan data prefetch and removal.

Because the high-performance storage acts as a cache, data that is marked as ready to be evicted should stay on the cache if the space is not needed for other jobs. This action helps save caching resources and time if a job that needs the same data starts later. A system overseeing this process is not part of this PoC, but might be realized with add-ons in existing workload and process managers.



Planning for Data Accelerator for AI and Analytics

This chapter provides information about planning for the different components for Data Accelerator for AI and Analytics (DAAA).

4.1 Security and data access rights considerations

Planning for a DAAA solution requires a security and data access design.

As use cases and needs vary greatly, we provide an overview and some pointers based on our proof of concept (PoC) system as examples to give you an idea of what you need.

In Figure 4-1, data is accessed in multiple storage systems by different users. APIs and applications are accessed by users.

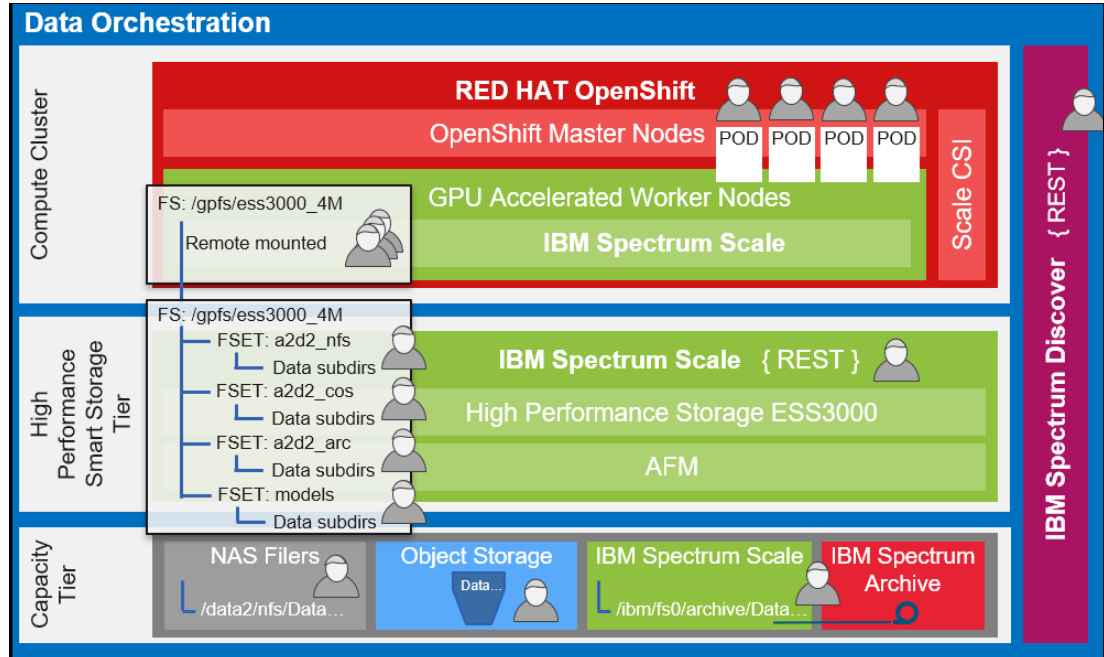


Figure 4-1 Showing our proof of concept data layout and users accessing data and applications

4.2 Data layer

The following sections provide more information about DAAA planning for the components that are part of the data layer.

4.2.1 Network-attached storage (NSF) Filer

Consider your security and data access concept (see 4.1, "Security and data access rights considerations" on page 28) when planning your Network File System (NFS) share.

For more information about, for example, Kerberos or access modes for IBM Spectrum Scale Active File Management (AFM) by using the NFSv3 protocol for the communication between clusters, see [IBM Knowledge Center](#).

4.2.2 Cloud object storage

Object storage is a computer data storage architecture that manages data as objects, as opposed to other architectures like file systems, which manage data as a file hierarchy, and Block Storage, which manages data as blocks within sectors and tracks.

The IBM Cloud Object Storage system hardware is composed of three servers: a Manager, IBM Accesser®, and IBM Slicestor®. When combined with IBM Cloud Object Storage system software, these servers make up an IBM Cloud Object Storage system. The Manager and Accesser server are 1U units, and Slicestor server is either 2U, 4U, or 5U units. Each of the servers contains a CPU, memory, network interface controller, disk storage controller, and the disk drives that are required for the unit. Each of these components is sold as a bare metal server, and with the addition of the ClevOS operating system, become an appliance that is used to deliver the IBM Cloud Object Storage system.

An IBM Cloud Object Storage system is a software-defined system that is also hardware aware, which means that although the ClevOS operating system can run on any standard x86-based server, it will not run to its full potential on hardware that has not been evaluated and certified by IBM to run ClevOS. Servers that are certified to run ClevOS take full advantage of this hardware awareness by ensuring that the server performs optimally from a monitoring, management, and performance perspective.

Planning host sizes and numbers of instances

IBM Cloud Object Storage supports an unlimited number of remote sites, although a typical deployment is a three-site geo-dispersed system. With a properly configured Information Dispersal Algorithm (IDA), one site can be lost to a disaster or network outage while the system continues to ingest and retrieve objects normally on the remaining two functioning sites. After the failed site is restored, the IBM Cloud Object Storage system automatically rebuilds the missing slices that were created while the site was down.

No single node has all of the data. This configuration makes it safe and less susceptible to data breaches while needing only a subset of the storage nodes to be available to retrieve the stored data. This ability to reassemble all the data from a subset of the slices dramatically increases the tolerance to node and disk failures.

Regarding the available bandwidth between sites for a multi-site deployment, you must have enough bandwidth to support the writes to the remote sites, or it is possible over time that the system becomes unbalanced, the sites that are lacking adequate bandwidth fall behind, and slices are written to the slow remote site. If this scenario happens intermittently, the rebuilder recognizes the missing slices and rebuilds them. If the situation is persistent, then the slow site never rebuilds the missing slices, and the stored data is never written to all the Slicestor servers in the system.

To determine the required bandwidth, you must know the required I/O performance of the system and the IDA. With this information, an IBM Cloud Object Storage architect can help calculate the required bandwidth for each remote site.

Users and security

Users interact with the IBM Cloud Object Storage system through the Amazon Simple Storage Service (Amazon S3) API standard.

Each IBM Cloud Object Storage system can have multiple sets of credentials, which consist of an Access Key ID and a Secret Access Key to provide access to the storage vaults (buckets) within the system. These credentials are required when enabling Amazon S3 API access from the application to the IBM Cloud Object Storage storage vaults. These keys plus the endpoint name (Accesser or load balancer server) are required to integrate the IBM Cloud Object Storage system with the AFM Amazon S3 gateway interface.

For more information about how to set users or access modes for AFM by using the Object Amazon S3 protocol for the communication between clusters, see [IBM Knowledge Center](#).

4.2.3 IBM Spectrum Archive Enterprise Edition Tape

IBM Spectrum Archive Enterprise Edition provides organizations an easy way to use cost-effective IBM tape drives and libraries within a tiered storage infrastructure. By using tape libraries instead of disks for data that is stored for long-term retention, organizations can improve efficiency and reduce costs. IBM Spectrum Archive Enterprise Edition seamlessly integrates with the scalability, resilience, and performance of IBM Spectrum Scale^{1, 2}.

IBM Spectrum Archive Enterprise Edition is based on the LTFS standard that is maintained by the Linear Tape Open (LTO) consortium and Storage Network and Industry Association (SNIA)³. IBM Spectrum Archive Enterprise Edition enhances the capabilities of the common LTFS editions (single drive edition and library edition⁴) by providing advanced tape management functions for a scaling environment.

IBM Spectrum Archive Enterprise Edition extends the IBM Spectrum Scale file systems as an external pool so that files can be seamlessly migrated from the disk tier to LTFS tape while providing transparent access to migrated files for the users. The migration is fully automated by using the IBM Spectrum Scale policy engine⁵. After a file is migrated to LTFS tape, it remains visible to the user in the IBM Spectrum Scale file system namespace. When the user accesses the file, IBM Spectrum Archive Enterprise Edition recalls it from the tape and copies it back to the IBM Spectrum Scale file system before the file can be opened by the user. This operation can take a few seconds to a couple of minutes because the tape must be mounted and positioned.

For more information about, for example, Kerberos or access modes for AFM by using the Network Shared Disk (NSD) protocol for the communication between clusters, see [IBM Knowledge Center](#).

Characteristics

IBM Spectrum Archive Enterprise Edition scales with the IBM Spectrum Scale cluster. A subset of IBM Spectrum Scale cluster servers can be configured as IBM Spectrum Archive Enterprise Edition enabled servers. Each IBM Spectrum Archive Enterprise Edition enabled server has tape drives that are attached where all tape drives must be installed in the same tape library. IBM Spectrum Archive Enterprise Edition operations such as migrations, recalls, reclamation, and reconciliation are distributed across all IBM Spectrum Archive Enterprise Edition enabled servers by the workload manager, allowing for optimal resource utilization.

All IBM Spectrum Archive Enterprise Edition enabled servers share tape cartridges in the tape library. Adding an IBM Spectrum Archive Enterprise Edition enabled server is seamless and allows the archive solution to scale dynamically.

Migrations and recalls are transparent to the file system user. The migration can be fully automated in a scalable environment by using the IBM Spectrum Scale policy engine⁵. After migration, the file is still visible in the file system namespace, and the user can seamlessly access the file. Upon file access, the file is recalled from tape. After the file is copied from tape to disk, the user can process the file.

¹ [IBM Spectrum Archive Enterprise Edition at IBM Knowledge Center](#)

² *IBM Spectrum Archive Enterprise Edition V1.3.0.6: Installation and Configuration Guide*, SG24-8333

³ [SNIA LTFS Format Specification](#)

⁴ [IBM Spectrum Archive Editions](#)

⁵ [IBM Spectrum Scale ILM Policy Guide for IBM Spectrum Archive Enterprise Edition](#)

Files can also be pre-migrated so that files are not moved from disk to tape but copied. After pre-migration, the file is dual-resident on disk and on tape. Subsequent migration of pre-migrated files does not have to copy the files to tape again, but can delete the file data in the IBM Spectrum Scale file system and create a stub file. This action allows for fast migrations in situations where the file system is overfilled. Pre-migration with IBM Spectrum Archive Enterprise Edition is not a backup solution because if a file in the IBM Spectrum Scale file system is deleted, it cannot be easily restored from LTFS.

When many files must be recalled within a short period, IBM Spectrum Archive Enterprise Edition offers the tape-optimized bulk recall function. With the bulk recall function, the administrator creates a list of files to be recalled and passes it to the `eeadm recall` command. IBM Spectrum Archive Enterprise Edition sorts the files by the tape-ID and the position on tape before it sequentially recalls all files that are on one tape.

One of the key advantages of LTFS is that it provides a standardized file system on tape, making the use of tapes much simpler. An LTFS tape is self-describing because it includes the file system index (directory structure), and the files are stored on the same tape cartridge. IBM Spectrum Archive Enterprise Edition creates the same IBM Spectrum Scale file and directory structure on the LTFS tapes, which fosters the interchangeability of data on tape instead of using slow and unreliable WAN connections. Therefore, IBM Spectrum Archive Enterprise Edition supports exporting and importing tapes.

4.3 High-performance storage with smart data cache layer

The following sections provide more information about DAAA planning for the components that are part of the high-performance storage with smart data cache layer.

4.3.1 IBM ESS 3000 and IBM Spectrum Scale

For IBM ESS 3000 and IBM Spectrum Scale, besides the standard planning considerations, no special planning guidance is needed in a DAAA environment.

You can find these standard planning considerations at IBM Knowledge Center:

- ▶ [IBM ESS 3000](#)
- ▶ [IBM Spectrum Scale](#)

Note: Because the DAAA PoC also includes the AFM to cloud object storage (COS) relationship, we had to patch the IBM ESS 3000 storage system with IBM Spectrum Scale V5.1. It is possible to upgrade the core scale packages, but this action puts your IBM ESS 3000 storage system into an unsupported state. At the time of writing, it is planned to provide IBM Spectrum Scale V5.1 on IBM ESS 3000 by the end of 1Q 2021. Nevertheless, existing IBM ESS 3000 storage systems also can be used with the DAAA solution when working with network-attached storage (NAS) Filers and other IBM Spectrum Scale storage architectures as AFM Home Site only.

The following section provides details to consider during the planning phase for the IBM Spectrum Scale AFM feature.

Active File Management

IBM Spectrum Scale AFM is a scalable and high-performance file system caching layer that is integrated within the IBM Spectrum Scale cluster file system. With AFM, you can create associations between IBM Spectrum Scale clusters, between IBM Spectrum Scale clusters and NFS data sources, or between IBM Spectrum Scale clusters and object data sources. AFM automates the flow of the data in asynchronous fashion. AFM with its automated flow of the data in asynchronous fashion helps implement a single namespace view across the sites, as shown in Figure 4-2.

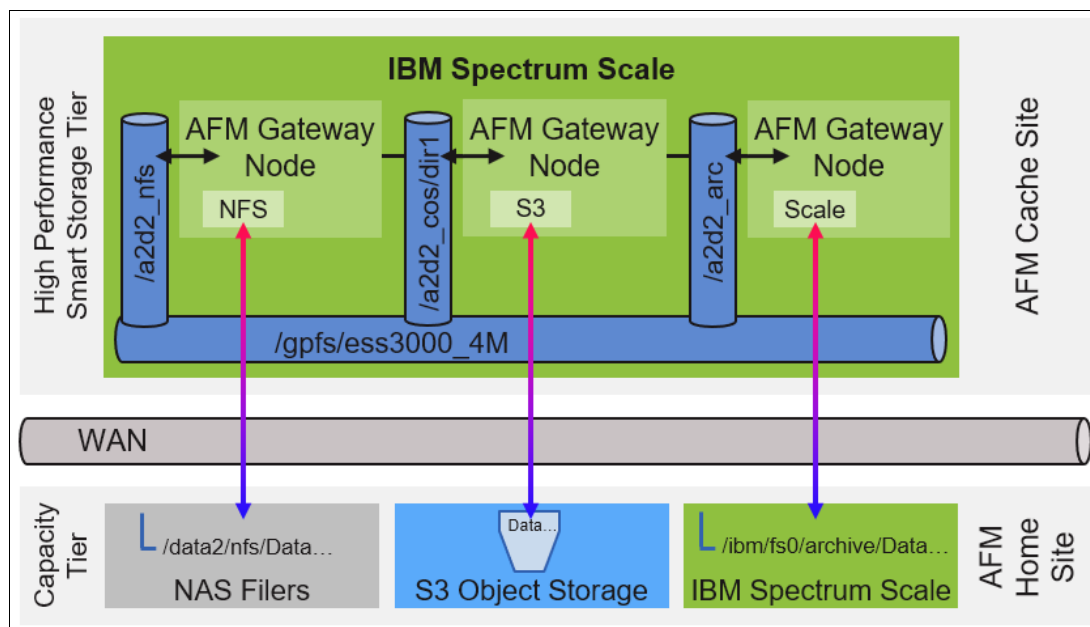


Figure 4-2 Implementing a single namespace view across the sites

Gateway node

Each AFM cache file set in a cluster is served by one of the nodes that is designated as a gateway node role in the cluster. The gateway node that is assigned to an AFM file set is called the *primary gateway* of the file set. The primary gateway acts as the owner of the file set, and it is responsible for all the communication with the home cluster. The primary gateway can also be configured to take help from other gateway nodes for parallel data synchronization for large files with the home site if the home site is an IBM Spectrum Scale cluster. This situation helps improve performance when synchronizing large files.

For more information about the primary gateway node, see [IBM Knowledge Center](#).

For more information about AFM gateway node planning, see [IBM Knowledge Center](#).

The AFM gateway node is licensed as a server node. The AFM gateway node role can be assigned only to a Linux member node that is available in the IBM Spectrum Scale cluster.

For more information about the functional support matrices for AFM, see [IBM Knowledge Center](#).

Active File Management to cloud object storage

The IBM Spectrum Scale AFM to COS feature enables caching of objects from the bucket in the COS server to the AFM file set. COS services such as Amazon S3 and IBM Cloud Object Storage offer industry-leading scalability, data availability, security, and performance. The AFM to COS feature enables associating a bucket at the COS server to the IBM Spectrum Scale AFM file set, or if configured to a directory, in the file set. AFM uses a bucket as the target of the file set or directory in the file set and synchronizes object data between them by using the HTTP and HTTPS protocols and internally calling the object storage APIs. All AFM modes are supported by this feature.

AFM supports two major use cases for data synchronization with a bucket: ObjectFS and ObjectOnly mode.

- **ObjectFS**

An AFM file set that is configured for ObjectFS mode behaves the same as an AFM file set. Revalidation is triggered periodically or on demand, and metadata is refreshed by a cache reflecting modification that is performed inside the bucket at the COS server. AFM auto-synchronizes modified objects' metadata in the bucket to the AFM COS ObjectFS file set.

- **ObjectOnly**

With ObjectOnly mode, changes that are performed at the bucket are not reflected automatically to the AFM cloud to object storage file set. This mode requires manual download of the data and metadata from the bucket to the AFM file set, which enables better utilization of resources and caters to the use case where a user wants to download data and perform modification at the AFM cache file set, and upload it back to the bucket.

The two behaviors (ObjectFS and Objectonly) are determined at the time of the AFM to COS file set creation, and they use AFM mode. In both cases, uploading the data, that is, the synchronization of changed data that originated from the AFM COS writable file set to the target bucket, is performed automatically without any user intervention. This behavior is the same as the AFM writable mode file set, and it is seamless to the user. This mode provides the most flexibility and control over uploads and download.

A user workspace directory can be created inside an AFM to COS mode file set and assigned to users. A directory inside an AFM COS file set can be mapped to a different bucket at the same or different COS server. A non-root user can perform uploads and downloads of data that belongs to their assigned workspace inside the AFM to COS file set. The directory-mapping feature enables mapping of a bucket to the directory inside the AFM file set.

For more information about AFM to COS, see [IBM Knowledge Center](#).

4.4 Compute cluster layer

The following sections provide more information about DAAA planning for the components that are part of the data compute cluster layer.

4.4.1 IBM Spectrum LSF

If deploying only a workload scheduler does not meet your needs, IBM Spectrum Load Sharing Facility (LSF) Suites can help meet the range of needs that a high-performance computing (HPC) environment faces. For users, it can take advantage of technologies such as accelerators to speed results. For the infrastructure, it can help get the most from more compute capacity that is available in the cloud during peaks in workloads. For the organization, it can help respond to market changes that drive constantly shifting priorities for both individual projects and the overall business. At the same time, it can help reduce costs and increase return on investment (ROI).

IBM Spectrum LSF Suites is available in three versions with progressively stronger capabilities: Workgroup, HPC, and Enterprise.

For more information about IBM Spectrum LSF Suites and planning, see [IBM Spectrum LSF Suites](#).

IBM Spectrum LSF provides a connector for Kubernetes that integrates the core IBM Spectrum LSF (LSF) scheduling technology into Kubernetes, which is described at [IBM Knowledge Center](#).

For our DAAA solution PoC, we used IBM Spectrum LSF and IBM Spectrum LSF Application Center V10.2 as one of the workload triggers that are described in 3.3, “Workload manager starts an analytics job” on page 24.

For the usage in our PoC, besides the standard planning considerations, no special planning guidance is needed in a DAAA environment.

4.4.2 Compute cluster

In more traditional environments like HPC, the compute nodes in the compute cluster typically are managed as available compute resources for queued batch workloads with a workload scheduler like, for example, LSF or Slurm. In containerized environments, the nodes are managed by a container orchestration platform like Kubernetes or OpenShift. In this use case, the native scheduling is more lightweight and aimed at running workloads interactively.

The number of compute nodes (or worker nodes in Kubernetes or OpenShift) and the selected node hardware (for example, CPUs, GPUs, memory, and network) heavily depend on the type and resource demands of the individual workloads, the number of users and batch jobs that run in parallel, and general time constraints. Running more batch jobs in parallel scales with the number of nodes. Reducing job run times for getting results faster can be met by scaling up (using more CPUs, GPUs, and memory per server) or scaling out, depending on the specific workloads and their ability to scale out horizontally. For example, the number of GPU resources in a single server is typically limited (such as PCIe slots), so large training workloads for deep neural networks (DNNs), which might run for weeks or even months, can be accelerated only by scaling out horizontally across more servers with GPU resources.

All compute nodes that are running workloads on data in IBM Spectrum Scale must be added as IBM Spectrum Scale client nodes to a local IBM Spectrum Scale cluster. Using the IBM Spectrum Scale NSD protocol, each one of these compute nodes has direct access to the IBM Spectrum Scale file systems and data in them. For high-bandwidth and low latency data access, a high-speed network like 40 Gbps or 100 Gbps Ethernet or EDR or HDR InfiniBand with RDMA is recommended. Planning considerations for IBM Spectrum Scale networking involve the *admin* network and the daemon network for daemon communication and data transfers. For more information, see [GPFS and network communication](#).

A minimum configuration of an OpenShift (Version 4 or higher) cluster starts with three *master* nodes and two *worker* nodes. The master nodes run the control plane and services that are essential to control the Kubernetes cluster. The worker nodes are the compute nodes that run the containerized workloads. The number of worker nodes must be scaled. The worker nodes advertise their capacities (CPUs, GPUs, memory, and other capacities), and the Kubernetes scheduler (as part of the master services) determines on which nodes to start the related containers and Pods for a workload deployment. For more information about the Red Hat OpenShift V4 architecture, see [OpenShift Container Platform architecture](#).

User and security context

A user typically interacts with the compute cluster through a scheduler and defines the workload to run. In traditional environments like HPC, this workload is most likely a defined batch job that is scheduled by a scheduler like LSF or Slurm and runs in the user's context with the user's *user ID* and *group ID* on the compute nodes. The user and group permissions and extended attributes in the IBM Spectrum Scale file system must be managed to control data access permissions and security. When requesting specific data through the DAAA workflow to be prefetched and made available in the high-performance tier, correct file access permissions must be considered so that the user can access the data, especially if the requested data might be retrieved from different storage architectures and locations.

In Kubernetes or OpenShift, a user typically defines and applies YAML manifests, which describe the required Kubernetes resources like Pods and persistent volume claims (PVCs) to run the workload, or, alternatively, run applications directly from the OpenShift web console. These resources are typically bound to the user's namespace (also referred to as a *project* in OpenShift).

A user's workload in OpenShift is running in the context of a Pod and its containers. Pods are the smallest unit of computing that can be deployed and managed in OpenShift. A Pod is composed of a group of one or more containers with shared storage or network resources and a specification about how to run the workload in the containers. Data access to persistent storage is provided through *persistent volumes* (PVs), which abstract the underlying physical storage layer and are mounted into the Pod's containers at a selected mount point. A PV is bound to the user's namespace through a successful PVC. It is tied to the user's namespace (or project) until the user deletes the claim. It cannot be bound again by another claim in another namespace. If the underlying data must be accessed by multiple projects in other namespaces, then each project needs its own PV, which can point to the same physical storage.

Regular users in OpenShift are not necessarily associated with a unique (fixed) user ID or group ID for accessing files in the PVs when running a Pod to run a workload. The processes in a Pod are typically running under an arbitrary user ID from a range of available IDs if nothing else is specified in the Pod specifications or the global environment.

So, special considerations apply when managing file permissions and data access for users and their containerized applications in OpenShift. You might need coordination between the cluster admin, the storage admin, and the user. The *cluster admin* creates *storage classes* for dynamic provisioning of new volumes and PVs through static provisioning to provide access to data in IBM Spectrum Scale. The *user* creates Pods and PVCs that bind to PVs to run containerized workloads.

OpenShift applies strict security standards. Users interacting with OpenShift Container Platform (Red Hat OCP) must first successfully authenticate to the OpenShift cluster. OpenShift allows for various identity providers (for example, LDAP) to be used to *authenticate* a user. Through role-based access control (RBAC) objects like *rules*, *roles*, and *role bindings*, OpenShift determines whether a user is *authorized* to perform an action within a project (or namespace). For more information, see [Understanding authentication](#).

In addition, *Security Context Constraints* (SCCs) define a set of conditions with which a Pod must comply. Pods eventually run the user's workload, and SCCs control the permissions for these Pods and determine the actions that they (and their collections of containers) can perform. For more information, see [Managing Security Context Constraints](#).

A regular user in OpenShift typically runs under the restricted SCC, which ensures that a Pod has the following considerations:

- ▶ It cannot run as privileged.
- ▶ It cannot mount host directory volumes.
- ▶ It must run as a user in a pre-allocated range of UIDs.
- ▶ It must run with a pre-allocated Multi-Category Security (MCS) label. MCS is an enhancement to SELinux that allows users to label files with categories.
- ▶ It may use any FSGroup.
- ▶ It may use any supplemental group.

SCCs are composed of settings and strategies that control the security features that a Pod can access.

The restricted SCC in Red Hat OpenShift V4.5.9 specifically defines the following strategies, which control access to volumes:

```
fsGroup:
  type: MustRunAs
runAsUser:
  type: MustRunAsRange
seLinuxContext:
  type: MustRunAs
supplementalGroups:
  type: RunAsAny
```

RunAsAny allows any ID (within a reasonable range that is supported by the image), and SCC strategies with MustRunAs or MustRunAsRange trigger ID validation and cause default values to be supplied by the Red Hat OCP to the container when these values are not supplied directly in the Pod definition or image. For more information about SCC strategies, see [SCC Strategies](#).

Managing access to data in containerized environments involves considering the file permissions (for example, user ID and group ID) in the back-end file system and SELinux settings, and ensuring that these IDs are allowed in the range of legal IDs that is defined for the project and namespace under the context of the SCC that matches the requirements of the Pod. SCCs influence whether a Pod is given a default user ID, fsGroup ID, supplemental group ID, and SELinux label, or if any IDs that are supplied in the Pod definition are acknowledged or fail (which means that the Pod also fails to start).

Within the limits of the applicable SCC, specific values can be defined in the `SecurityContext` section of a Pod to allow more granular control of access permissions to data in PVs, for example:

```
spec:
  securityContext:
    runAsUser: 1000611000
    runAsGroup: 1000622000
    supplementalGroups: [100]
```

For more information, see [Configure a Security Context for a Pod or Container](#) and [Volume Security - Supplemental Groups](#).

In the example in this section, the `runAsUser` and `runAsGroup` fields specify that all processes in the Pod's containers run with user ID 1000611000 and the primary group ID of 1000622000 plus the supplemental group ID 100 (the group *users* on Red Hat Enterprise Linux (RHEL) systems). Any files that are created are owned by user ID 1000611000 and group ID 1000622000. If `runAsGroup` is omitted (which is generally the case), then the group ID defaults to 0 (*root* group). This situation might be required to access the mount points of PVs. The above example is meant to show how user IDs and group IDs and file permissions can work together in containerized environments. In general, a user does not specify user and group IDs for the Pods.

4.5 Data catalog layer

The following sections provide more information about DAAA planning for the components that are part of the data catalog layer.

4.5.1 IBM Spectrum Discover

IBM Spectrum Discover is a modern metadata management software that provides data insight for exabyte-scale heterogeneous file, object, backup, and archive storage on-premises and in the cloud. The software easily connects to these data sources to rapidly ingest, consolidate, and index metadata for billions of files and objects. IBM Spectrum Discover catalogs all system metadata initially. This catalog can be further enriched with both custom and derived metadata. The IBM Spectrum Discover catalog can then be used to quickly refine the data set for the artificial intelligence (AI) workload by using various facets of the metadata.

Deployment options

IBM Spectrum Discover can either be deployed as a virtual appliance or as an application on the Red Hat OCP. For more information, see the “Planning” section of the IBM Spectrum Discover [IBM Knowledge Center](#).

Data source connections

IBM Spectrum Discover requires a data source connection to both the performance tier and capacity tier to index all available file and object system metadata. For more information about creating data source connections in IBM Spectrum Discover, see [IBM Knowledge Center](#).

To get the latest metadata from a data source, initiate a scan from IBM Spectrum Discover. During the data source scan, IBM Spectrum Discover interrogates the source storage system for all file and object system metadata. Optionally, both IBM Spectrum Scale and IBM Cloud Object Storage support sending metadata updates directly to IBM Spectrum Discover by using the built-in messaging service. In this mode, the IBM Spectrum Discover catalog always reflects the most current metadata.

For more information about enabling IBM Spectrum Scale live events, see [IBM Knowledge Center](#).

For more information about enabling IBM Cloud Object Storage notifications, see [IBM Knowledge Center](#).

Built-in data mover

IBM Spectrum Discover offers a built-in data mover application known as the ScaleAFM application, which runs the AFM function within IBM Spectrum Scale. Using the ScaleAFM application offers a couple of advantages:

- ▶ Guided visual window to help reduce the complexity of using CLI tools.
- ▶ Ability to use all metadata within the IBM Spectrum Discover catalog to make informed decisions about which files and objects to copy or move.

For more information, see [IBM Knowledge Center](#).

Because our PoC includes multiple different capacity tier storage systems, we use the IBM Spectrum Discover search REST interface to create the file lists that are needed to cache or evict files by running a script in IBM Spectrum Scale AFM, and provide the lists to AFM directly. For sample scripts, see Appendix A, “Code samples” on page 71.

Tag ingest

IBM Spectrum Discover can ingest a set of tags, or labels, from an externally curated source. This function can be especially useful when dealing with portable data sets that consist of data and curated metadata. By ingesting the custom or derived metadata of the portable data set, this information then can be used when defining or refining a data set of an AI workload.

For more information, see [IBM Knowledge Center](#).

Users

IBM Spectrum Discover offers various roles to restrict what certain users may do and see. Due to their impact on the source storage systems and the IBM Spectrum Discover catalog, both tag ingest and data movement require administrator privileges. The user performing these operations must have the Data Admin role assigned.

For more information, see [IBM Knowledge Center](#).



Deployment considerations for Data Accelerator for AI and Analytics

This chapter describes deployment help for the different components that are part of Data Accelerator for AI and Analytics (DAAA).

5.1 Data layer

The following sections provide more information about DAAA deployment considerations for the components that are part of the data layer.

5.1.1 Network-attached storage (NAS) Filer

In addition to considering the security and data access concept that is described in 4.1, “Security and data access rights considerations” on page 28, you should review the IBM Spectrum Scale Active File Management (AFM) home cluster configuration details at [IBM Knowledge Center](#).

Configuring NAS Filer in the proof of concept environment

In our proof of concept (PoC) environment, we use a node with IP address 192.168.1.240 as the Network File System (NFS) Server host with the following settings for export:

```
/data2/nfs
192.168.1.0/24(rw,nohide,insecure,no_subtree_check,sync,no_wdelay,no_root_squash,fsid=105)
```

To review the IP addresses, see Figure 2-2 on page 8.

5.1.2 IBM Cloud Object Storage

This section describes best practices when deploying IBM Cloud Object Storage. They are the critical design decision points when designing an IBM Cloud Object Storage system.

For more information about the IBM Cloud Object Storage configuration, see 2.2, “Proof of concept environment” on page 14.

Best practices and critical design decision points

The following sections provide best practices and critical design decisions points for the Information Dispersal Algorithm (IDA), dispersal modes, and the storage vault.

IDA

The IDA is based on erasure coding and defines the reliability, availability, and storage efficiency of an IBM Cloud Object Storage system. The IDA is defined at the vault level at the vault creation time. The IDA is written as *width/read threshold/write threshold*, for example, 12/6/8.

If the read threshold is set higher, the IBM Cloud Object Storage system can survive fewer failures, but the storage efficiency is better.

If the write threshold is set lower, the IBM Cloud Object Storage system can survive more failures, but the storage efficiency suffers because of the higher redundancies.

Dispersal modes

IBM Cloud Object Storage can operate in two different dispersal modes: standard dispersal (SD) mode and concentrated dispersal (CD) mode:

- ▶ SD Mode configures width, read, and write thresholds.
- ▶ CD Mode defines preconfigured IDAs that are optimized for storage or performance.

The two modes have three location options: Single site, two sites mirrored, or geo-dispersed multiple sites (most common).

Storage mode

The default for an IBM Cloud Object Storage system is *vault mode*, which is suitable for most customer deployments. For deployments that support thousands of buckets or tenants, IBM Cloud Object Storage can be deployed in container mode.

The general term for a logical storage unit in Amazon Simple Storage Service (Amazon S3) is a *bucket*. In vault mode, a bucket is referred to as a vault. In container mode, a bucket is referred to as a container.

Configuring IBM Cloud Object Storage in the proof of concept environment

In our PoC environment, we use an IBM Cloud Object Storage on-premises solution running in virtual machines (VMs) as an object storage solution. For more information about how to configure an IBM Cloud Object Storage system, see [IBM Knowledge Center](#).

In the following section, we describe only the IBM Cloud Object Storage configuration details for the DAAA and our PoC. Here are the steps that must be completed:

1. Configure a single Accesser (10.10.1.211), a single Manager (10.10.1.210), and three SliceStor (10.10.1.212-4) devices.

Two vaults are created:

- A vault with the name *a2d2* holds the sample data set data.
- A vault with the name *Models* receives data that is written by the sample workload.

2. Add a notification service to both vaults:

Name: Discover Notifications (Pool Default)
Topic: cos-1e-connector-topic

3. Create an account that is named BDA that has a user that is named bda. The user has owner vault access to the a2d2 and Models vault. To decide which vault access is needed, see 4.1, “Security and data access rights considerations” on page 28.
4. IBM Spectrum Discover provides automatic metadata catalog updates through a notification service. Notification is also supported by IBM Cloud Object Storage. For more information, see [IBM Knowledge Center](#).

To configure the notification service, see [IBM Knowledge Center](#).

In our PoC, we used the following configuration:

Name: Discover Notifications
Topic: cos-1e-connector-topic
Hosts: 192.168.1.230:9092
Type: IBM Spectrum Discover

5. Enable authentication and encryption. For more information about how to gather the needed details see the links in step 4.

5.1.3 IBM Spectrum Archive Enterprise Edition Tape

IBM Spectrum Archive Enterprise Edition is installed on one or more servers that are members of an IBM Spectrum Scale cluster. These servers have the tape drives and libraries attached that are used by IBM Spectrum Archive.

The installation of IBM Spectrum Archive Enterprise Edition is described in [IBM Knowledge Center](#).

Configuring IBM Spectrum Archive Enterprise Edition in the proof of concept environment

In our PoC environment, we use IBM Spectrum Archive Enterprise Edition V1.3.0.7 that is installed on one server that is a member of an IBM Spectrum Scale cluster.

In addition to the standard installation and configuration, we configured a new user *archi* with only the privileges to run migrate and recall operations remotely through SSH. The recall and migrate calls are run as the *archi* user.

The used file system is named *fs0* and mounted on path */ibm/fs0*. This file system has one internal system pool. There is a file set with the name *archive* that is configured on path */ibm/fs0/archive*. All files that are subject to migration to tape or will be recalled from tape are in this file set.

IBM Spectrum Archive is configured on one node that is named *archive* (192.168.1.240), and there are four tape drives that are connected to this node.

To recall a list of files from tape, run the following command:

```
sudo eeadm recall <list of files to be recalled>
```

To migrate a list of files to tape, run the following command:

```
sudo eeadm migrate <list of files to be migrated> -p pool1
```

5.2 High-performance storage with smart data cache layer

The following sections provide more information about DAAA deployment considerations for the components that are part of the high-performance storage with smart data cache layer.

5.2.1 IBM ESS 3000 and IBM Spectrum Scale

For IBM ESS 3000 and IBM Spectrum Scale, there is no special deployment guidance that is needed in a DAAA environment aside from standard considerations:

- ▶ [IBM ESS 3000](#)
- ▶ IBM Spectrum Scale:
 - [Installing](#)
 - [Configuring](#)
 - [Administering](#)

The following section provides details that should be considered during the deployment phase for the IBM Spectrum Scale AFM feature.

For more information about the general IBM Spectrum Scale AFM commands, see [IBM Knowledge Center](#).

Active File Management to cloud object storage

The AFM to cloud object storage (COS) file set uses a bucket or vault in the object storage server. AFM uses this bucket as the `afmTarget` and synchronize both sites by using either the HTTP or HTTPS protocol. The AFM to COS file set can be configured either as ObjectFS or ObjectOnly mode. Determining which mode is suitable is based on the workload requirement.

Active File Management to cloud object storage administration commands

Administration of the AFM to COS file set is controlled by the newly added commands `mmafmcoskeys`, `mmafmcosconfig`, `mmafmcosctl`, and `mmafmcosaccess`, and existing AFM commands like `mmchfileset` and `mmafmctl`.

To perform downloads and uploads of data that is specific to a user's workspace, run the `mmafmcosctl` command, which provides upload, download, and evict operations with regards to objects at the cache file set.

For more information about the `mmafmcosctl` command, see [IBM Knowledge Center](#).

Users can map a directory inside an AFM to COS file set to a different bucket at the same or a different IBM Cloud Object Storage server. This action is supported by the `mmafmcosaccess` command.

For more information about the `mmafmcosaccess` command, see [IBM Knowledge Center](#).

Existing commands are still applicable to the AFM to COS file set for various administrations tasks. To check the status of an AFM to COS file set, run the `mmafmctl getstate` command.

A failover of the AFM to COS file set is the same as an AFM file set. To perform a failover of AFM to COS target bucket to another target bucket, run the `mmafmctl failover` command.

A modification of AFM to COS file set level parameters is performed by running the `mmchfileset` command.

Configuring an ObjectFS mode file set

ObjectFS mode can be configured by using the `--object-fs` parameter with the `mmafmcosconfig` command. Here is an example:

```
# mmafmcosconfig fs1 SW1 --endpoint http://<IP>:<port> --object-fs
--bucket bucket1 --mode sw
```

Configuring an ObjectOnly mode file set

ObjectOnly mode can be configured by running the `mmafmcosconfig` command. Here is an example:

```
# mmafmcosconfig fs1 SW1 --endpoint http://<IP>:<port> --bucket bucket1 --mode iw
```

Active File Management to cloud object storage parameters

Here are few of the parameters of the `mmafmcosconfig` command:

- | | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>--xattr</code> | Specifies user-extended attributes to be synchronized with a COS. If this option is not set, the AFM to COS file set does not synchronize user-extended attributes with the COS. |
| <code>--acls</code> | Enables the cache to synchronize access control lists (ACLs) to the COS server. If this option is not set, AFM does not synchronize ACLs to COS. The default is disabled ACL synchronization. |
| <code>--user-keys</code> | Specifies adding a callback for user keys. Users must place the <code>mmuidkeys</code> file under <code>/var/mmfs/etc/mmuid2keys</code> . |

--mode	All AFM file set modes support an AFM to COS file set. These modes are independent-writer (IW), single-writer (SW), read-only (RO), and local-updates (LU). The default is the SW mode.
--uid	Specifies a user ID to be set on an AFM to COS file set. If this option is not set, a default owner is set, for example, root.
--gid	Specifies a group ID to be set on an AFM to COS file set. If this option is not set, a default group is set, for example, root.

For more information about the `mmafmcscosconfig` command, see [IBM Knowledge Center](#).

Configuring Active File Management to cloud object storage in the proof of concept environment

In our PoC environment, we use an IBM ESS 3000 as our high-performance storage system and IBM Spectrum Scale AFM as our smart data cache. For more information about how to configure an IBM Spectrum Scale file system, see the IBM Spectrum Scale documentation. In the following section, only the necessary configuration details for only for DAAA and our PoC are described.

IBM ESS storage cluster

For more information about how to configure a IBM Spectrum Scale cluster and file system, see [Installing](#), [Configuring](#), and [Administering](#).

To configure the IBM ESS storage cluster for the PoC, complete the following steps:

1. Run the following command and look at the output that follows it:

```
# mmlscluster

GPFS cluster information
=====
GPFS cluster name:      ess3000.bda.scale.ibm.com
GPFS cluster id:       215057217487177715
GPFS UID domain:      ess3000.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     fsccl-fab3-3-a-priv.bda.scale.ibm.com 192.168.1.122 fsccl-fab3-3-a-priv.bda.scale.ibm.com
quorum-manager-perfmon
4     fsccl-fab3-3-b-priv.bda.scale.ibm.com 192.168.1.123 fsccl-fab3-3-b-priv.bda.scale.ibm.com
quorum-manager-perfmon
6     node12.bda.scale.ibm.com             192.168.1.52  node12.bda.scale.ibm.com
quorum-manager-gateway-perfmon
```

2. Create a file system that is named `ess3000_4M` with the following configuration:

```
# mmlsfs ess3000_4M
flag      value      description
-----
-f         8192      Minimum fragment (subblock) size in bytes
-i         4096      Inode size in bytes
-I         32768     Indirect block size in bytes
-m         1         Default number of metadata replicas
-M         2         Maximum number of metadata replicas
-r         1         Default number of data replicas
-R         2         Maximum number of data replicas
-j         scatter  Block allocation type
-D         nfs4      File locking semantics in effect
-k         all      ACL semantics in effect
-n         32         Estimated number of nodes that will mount file system
-B         4194304  Block size
-Q         user;group;fileset
           user;group;fileset  Quotas accounting enabled
           user;group;fileset  Quotas enforced
           none              Default quotas enabled
--perfilesset-quota yes      Per-filesset quota enforcement
--filessetdf yes      Filesset df enabled?
-V         24.00 (5.1.0.0)  Current file system version
           22.00 (5.0.4.0)  Original file system version
```



```

--create-time      Mon May 11 20:19:47 2020  File system creation time
-z                no                        Is DMAPi enabled?
-L                33554432                  Logfile size
-E                no                        Exact mtime mount option
-S                relatime                  Suppress atime mount option
-K                whenpossible              Strict replica allocation option
--fastea          yes                       Fast external attributes enabled?
--encryption      no                        Encryption enabled?
--inode-limit     19727808                  Maximum number of inodes in all inode spaces
--log-replicas    0                        Number of log replicas
--is4KAligned     yes                       is4KAligned?
--rapid-repair    yes                       rapidRepair enabled?
--write-cache-threshold 0                    HAWC Threshold (max 65536)
--subblocks-per-full-block 512              Number of subblocks per full block
-P                system                    Disk storage pools in file system
--file-audit-log   no                       File Audit Logging enabled?
--maintenance-mode no                      Maintenance Mode enabled?
-d                RG001LG001VS002;RG001LG002VS002;RG001LG003VS002;RG001LG004VS002  Disks in file system
-A                yes                       Automatic mount option
-o                none                       Additional mount options
-T                /gpfs/ess3000_4M          Default mount point
--mount-priority  0                        Mount priority

```

For the AFM with COS function, the required minimum file system version is Version 24.00 (5.1.0.0).

The file sets under the ess3000_4M file system are created by the AFM commands that are provided in “General Active File Management settings” on page 47, “Adding the NFS server relationship to Active File Management” on page 48, “Adding the IBM Cloud Object Storage relationship to Active File Management” on page 48, and “Adding the IBM Spectrum Scale storage / archive system relationship to AFM” on page 49.

3. After the AFM relationships are created, run the following command and look at its output, which should look like the following output:

```
# mmlsfileset ess3000_4M --afm -L
Filesets in file system 'ess3000_4M':
```

Attributes for fileset root:

=====

```

Status                Linked
Path                  /gpfs/ess3000_4M
Id                    0
Root inode            3
Parent Id             --
Created               Mon May 11 20:19:56 2020
Comment               root fileset
Inode space           0
Maximum number of inodes 15725568
Allocated inodes      790528
Permission change flag chmodAndSetacl
afm-associated        No

```

Attributes for fileset a2d2_nfs:

=====

```

Status                Linked
Path                  /gpfs/ess3000_4M/a2d2_nfs
Id                    1
Root inode            524291
Parent Id             0
Created               Fri Sep 25 08:54:20 2020
Comment
Inode space           1
Maximum number of inodes 1000448
Allocated inodes      507904

```

Permission change flag	chmodAndSetac1
afm-associated	Yes
Target	nfs://192.168.1.1/data2/nfs
Mode	independent-writer
File Lookup Refresh Interval	30 (default)
File Open Refresh Interval	30 (default)
Dir Lookup Refresh Interval	60 (default)
Dir Open Refresh Interval	60 (default)
Async Delay	15 (default)
Last pSnapId	0
Display Home Snapshots	no
Number of Gateway Flush Threads	4
Prefetch Threshold	0 (default)
Eviction Enabled	yes (default)
IO Flags	0x0 (default)

Attributes for fileset a2d2_cos:

=====

Status	Linked
Path	/gpfs/ess3000_4M/a2d2_cos
Id	3
Root inode	1048579
Parent Id	0
Created	Fri Oct 9 12:14:13 2020
Comment	
Inode space	2
Maximum number of inodes	1000448
Allocated inodes	507904
Permission change flag	chmodAndSetac1
afm-associated	Yes
Target	http://10.10.1.211:80/a2d2
Mode	independent-writer
File Lookup Refresh Interval	120
File Open Refresh Interval	120
Dir Lookup Refresh Interval	120
Dir Open Refresh Interval	120
Async Delay	15 (default)
Last pSnapId	0
Display Home Snapshots	no
Parallel Read Chunk Size	0
Number of Gateway Flush Threads	4
Prefetch Threshold	0 (default)
Eviction Enabled	yes (default)
Parallel Write Chunk Size	0
IO Flags	0x80000 (afmObjectXattr)

Attributes for fileset a2d2_arc:

=====

Status	Linked
Path	/gpfs/ess3000_4M/a2d2_arc
Id	7
Root inode	4194307
Parent Id	0
Created	Wed Oct 7 10:21:17 2020
Comment	

Inode space	4
Maximum number of inodes	1000448
Allocated inodes	507904
Permission change flag	chmodAndSetacl
afm-associated	Yes
Target	gpfs:///gpfs/archive/archive
Mode	independent-writer
File Lookup Refresh Interval	30 (default)
File Open Refresh Interval	30 (default)
Dir Lookup Refresh Interval	60 (default)
Dir Open Refresh Interval	60 (default)
Async Delay	15 (default)
Last pSnapId	0
Display Home Snapshots	no
Number of Gateway Flush Threads	4
Prefetch Threshold	0 (default)
Eviction Enabled	yes (default)
IO Flags	0x0 (default)

Attributes for fileset models:

=====

Status	Linked
Path	/gpfs/ess3000_4M/models
Id	9
Root inode	4718595
Parent Id	0
Created	Fri Oct 30 13:41:16 2020
Comment	
Inode space	5
Maximum number of inodes	1000448
Allocated inodes	507904
Permission change flag	chmodAndSetacl
afm-associated	Yes
Target	http://10.10.1.211:80/Models
Mode	independent-writer
File Lookup Refresh Interval	120
File Open Refresh Interval	120
Dir Lookup Refresh Interval	120
Dir Open Refresh Interval	120
Async Delay	15 (default)
Last pSnapId	0
Display Home Snapshots	no
Parallel Read Chunk Size	0
Number of Gateway Flush Threads	4
Prefetch Threshold	0 (default)
Eviction Enabled	yes (default)
Parallel Write Chunk Size	0
IO Flags	0x80000 (afmObjectXattr)

The following sections provide details about how the relationships to the NFS server, COS, and the scale storage / archive system were created.

General Active File Management settings

Run the following commands on the cache cluster. In our case, we used the management node of our IBM ESS storage cluster (192.168.1.52).

```
mm1scluster  
mmchnode --gateway -N node12.bda.scale.ibm.com
```

Adding the NFS server relationship to Active File Management

Ensure that the NFS server that is acting as the home server is configured as described in 5.1.1, “Network-attached storage (NAS) Filer” on page 40. To add an NFS server relationship on the cache cluster, see [IBM Knowledge Center](#).

To configure the created NFS share as an NFS relationship to our ess3000_4M file system (see “Configuring NAS Filer in the proof of concept environment” on page 40), we used the parameters adjust quota, afm mode, and other settings as needed by your security and data access concept, which is described in 4.1, “Security and data access rights considerations” on page 28.

Run the following commands on the cache cluster. In our case, we used the management node of our IBM ESS storage cluster (192.168.1.52).

```
mmcrfileset ess3000_4M a2d2_nfs -p afmtarget=192.168.1.1:/data2/nfs -p afmmode=iw  
--inode-space new --inode-limit 1000000  
mm1inkfileset ess3000_4M a2d2_nfs -J /gpfs/ess3000_4M/a2d2_nfs
```

To review the IP addresses that were used, see Figure 2-2 on page 8.

These commands provided the cached NFS share in our ess3000_4M file system under /gpfs/ess3000_4M/a2d2_nfs.

Adding the IBM Cloud Object Storage relationship to Active File Management

Ensure that the IBM Cloud Object Storage system that is acting as the Home Server is configured as described in 5.1.2, “IBM Cloud Object Storage” on page 40 by completing the following steps:

1. Read the “Introduction to AFM to COS” documentation in [Table 1. Links to various AFM to Cloud Object Storage help topics](#) to understand the provided features, operation modes, and limitations.
2. To add a bucket of an IBM Cloud Storage system relationship on the cache cluster, following the instructions at [IBM Knowledge Center](#).
3. For the next steps, the *Access Key ID* and *Secret Access Key* of the user that accesses the bucket in the IBM Cloud Object Storage system is needed. To gather those details, open the IBM Cloud Object Storage Manager GUI and go to the Security window. Select the user in the **Accounts** table. The account details for the user show up, and the section “Access Key Authentication” provides the needed key details, as shown in Figure 5-1 on page 49.

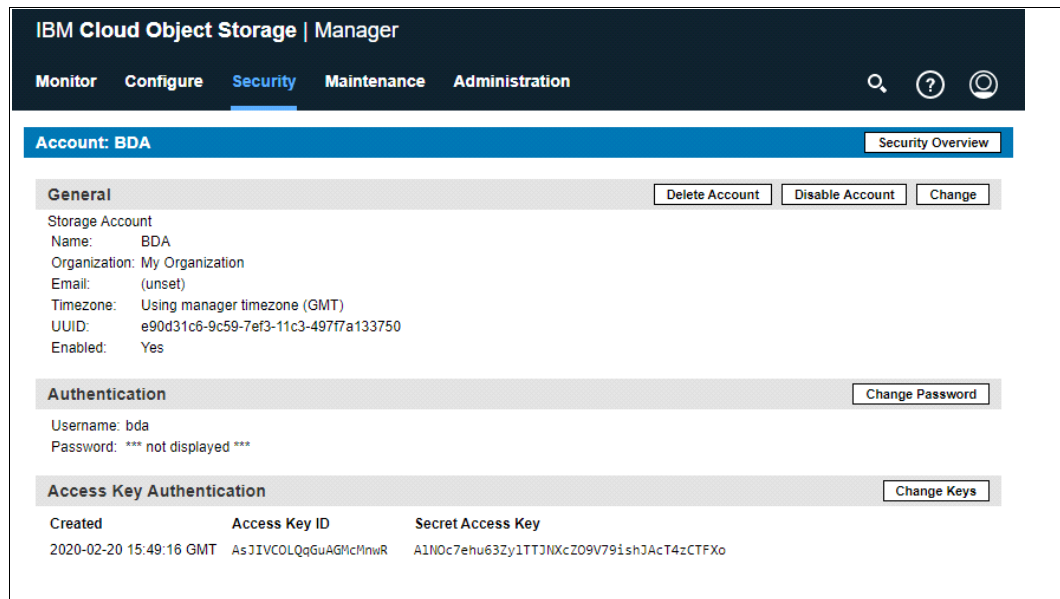


Figure 5-1 Gathering the needed key details from the Security window in the IBM Cloud Object Storage Manager GUI

4. To configure an IBM Cloud Object Storage vault or bucket that is named a2d2 as an object relationship to our ess3000_4M file system, we use the parameters adjust quota, permission, afm / object mode, and other settings as needed by your security and data access concept. For more information, see 4.1, “Security and data access rights considerations” on page 28.
5. Run the following commands on the cache cluster. In our case, we used the management node of our IBM ESS storage cluster (192.168.1.52).

- a. Create the relationship to the a2d2 bucket:

```
mmafmcosskeys a2d2 set AsJIVCOLQqGuAGMcMnwR
A1N0c7ehu63Zy1TTJNXcZ09V79ishJAcT4zCTFXo
```

```
mmafmcossconfig ess3000_4M a2d2_cos --endpoint http://10.10.1.211 --object-fs
--bucket a2d2 --quota-files 1000000 --cleanup --xattr --perm 555 --mode iw
```

- b. Create the relationship to the Models bucket:

```
mmafmcosskeys models set AsJIVCOLQqGuAGMcMnwR
A1N0c7ehu63Zy1TTJNXcZ09V79ishJAcT4zCTFXo
```

```
mmafmcossconfig ess3000_4M models --endpoint http://10.10.1.211 --object-fs
--bucket Models --quota-files 1000000 --cleanup --xattr --mode iw
```

To review the used IP addresses, see Figure 2-2 on page 8. The IP addresses use the 100 GbE data network and refer to the IBM Cloud Accesser node (192.168.1.211).

These commands provided the cached IBM Cloud Object Storage buckets in our ess3000_4M file system under /gpfs/ess3000_4M/a2d2_cos and /gpfs/ess3000_4M/models.

Adding the IBM Spectrum Scale storage / archive system relationship to AFM

Ensure that the IBM Spectrum Scale storage system that is connected to IBM Spectrum Archive Enterprise Edition that is acting as the home server is configured as described in 5.1.3, “IBM Spectrum Archive Enterprise Edition Tape” on page 41. To add an IBM Spectrum Scale Storage System relationship on the cache cluster, follow the instructions in that section.

Because a IBM Spectrum Scale home cluster exists and is configured with a file system and file set, create remote access by starting with step 4 in the instructions at [IBM Knowledge Center](#). In our PoC environment, we used the following commands:

- ▶ `mmauth genkey new`
- ▶ `mmauth update . -l AUTHONLY`
- ▶ `scp /var/mmfs/ssl/id_rsa.pub 192.168.1.52:/tmp/archive_id_rsa.pub`
- ▶ `mmauth genkey new`
- ▶ `mmauth update . -l AUTHONLY`
- ▶ `scp /var/mmfs/ssl/id_rsa.pub 192.168.1.240:/tmp/ess3000_id_rsa.pub`
- ▶ `mmauth add ess3000.bda.scale.ibm.com -k ess3000_id_rsa.pub`
- ▶ `mmauth grant ess3000.bda.scale.ibm.com -f fs0`
- ▶ `mmremotecluster add spectrumarchive.bda.scale.ibm.com -n archive`
- ▶ `mmremotefs add archive_4M -f fs0 -C spectrumarchive.bda.scale.ibm.com -T /gpfs/archive`
- ▶ `mmmount archive_4M`

Continue with step 5 in [IBM Knowledge Center](#), and use the following command:

```
mmafmconfig enable /ibm/fs0/archive
```

To create the AFM with IBM Spectrum Scale relationship, see [IBM Knowledge Center](#), and use the following commands:

- ▶ `mmcrfileset ess3000_4M a2d2_arc -p afmtarget=gpfs:///gpfs/archive/archive -p afmnode=iw --inode-space=new --inode-limit 1000000`
- ▶ `mmmlinkfileset ess3000_4M a2d2_arc -J /gpfs/ess3000_4M/a2d2_arc`

To see the IP addresses that we use, see Figure 2-2 on page 8.

These commands provided the cached IBM Spectrum Scale file set in our ess3000_4M file system under /gpfs/ess3000_4M/a2d2_arc.

5.3 Compute cluster layer

The following sections provide more details about DAAA deployment considerations for the components that are part of the compute cluster layer.

5.3.1 IBM Spectrum LSF

As described in 4.4.1, “IBM Spectrum LSF” on page 34, the IBM Spectrum Load Sharing Facility (LSF) Suite is available and provides three different versions. The following list contains links to the standard deployment guidance and considerations:

- ▶ Documentation:
 - [LSF](#)
 - [LSF Suites for Workgroups](#)
 - [LSF Suites High-Performance Computing \(HPC\)](#)
 - [LSF Suites Enterprise](#)

- Installation:
 - [LSF](#)
 - [LSF Suites for Workgroups](#)
 - [LSF Suites HPC](#)
 - [LSF Suites Enterprise](#)

Configuring IBM Spectrum LSF in the proof of concept environment

In our PoC environment, we use an IBM Spectrum LSF installation running in a VM. IBM Spectrum LSF is used as workload management platform to orchestrate the management and creation of the required resources in the DAAA workflow. Finally, schedule the workload on an OpenShift cluster.

In the following section, we provide a brief description of the IBM Spectrum LSF setup that was used for the PoC in this paper to showcase the DAAA workflow.

The LSF instance is configured as a kernel-based virtual machine (KVM) with four logical CPUs, 8 GB of memory, and a 30 GB system disk running CentOS7 (7.7-1908). The network interface was bridged, which grants full access to the physical admin network with IP address 192.168.1.200 (`lsf.bda.scale.ibm.com`).

We use LSF V10.1 Standard Edition with patch `lsf10.1_linux310-lib217-x86_64-532214.tar.Z`. In addition to an admin user (*lsfadmin* (ID 5000)), which is required for the installation, we also create a user *dean* (ID 2000) as an actual user to schedule jobs with LSF. IBM Spectrum LSF does not allow running jobs as root user. The user *dean* represents our data scientist persona. All jobs are run in this user's context.

LSF was installed on the local boot disk in the directory `/shared/lsf`, and the LSF daemon was enabled and started (`systemctl enable lsfd; systemctl start lsfd`).

In addition, we installed IBM Spectrum LSF Application Center V10.2 under `/opt/ibm/lsfsuite/ext`, which provides a flexible and easy to use interface for cluster users and administrators. It is an add-on module to IBM Spectrum LSF that enables users to interact with intuitive and self-documenting standardized interfaces. For more information, see the [IBM Spectrum LSF Application Center V10.2 documentation](#).

The successful installation of IBM Spectrum LSF Application Center is required to add the flag `ENABLE_EVENT_STREAM=y` to the `/shared/lsf/conf/lsbatch/cluster1/configdir/lsb.params` configuration file of IBM Spectrum LSF.

The IBM Spectrum LSF Application Center GUI is available at `http://lsf.bda.scale.ibm.com:8080` in our environment. With the GUI, you can create user-defined job submission templates. For more information, see Figure 3-5 on page 24.

This template can be created by using the LSF provided template builder. For a sample XML file, see Appendix A, “Code samples” on page 71.

5.3.2 IBM Spectrum Scale storage cluster

This section describes the deployment considerations for DAAA regarding IBM Spectrum Scale in the compute cluster layer. All nodes in the compute cluster that are running workloads on data in IBM Spectrum Scale (the high-performance smart storage tier) must be added as IBM Spectrum Scale client nodes to a local IBM Spectrum Scale storage cluster.

General considerations about the prerequisites and the deployment of IBM Spectrum Scale can be found in [IBM Knowledge Center](#).

Container orchestration platforms like OpenShift or Kubernetes that are running on the compute cluster introduce an extra layer of abstraction for the consumption of persistent storage. Access to data and persistent storage in IBM Spectrum Scale for containerized applications is provided through the IBM Spectrum Scale Container Storage Interface (CSI) driver.

General considerations about the prerequisites and the deployment of the IBM Spectrum Scale CSI Driver can be found in [IBM Knowledge Center](#).

Configuring IBM Spectrum Scale in the proof of concept environment

In the PoC setup, we use OpenShift V4.5.9 as container orchestration platform on the compute cluster, as shown in Figure 2-7 on page 15, for Red Hat OpenShift and IBM Spectrum Scale node roles and software releases. The GPU-accelerated OpenShift worker nodes accessing data in the high-performance smart storage tier through the DAAA pipeline also must be client nodes of the local IBM Spectrum Scale storage cluster.

IBM Spectrum Scale

To add the two GPU-accelerated and Red Hat Enterprise Linux (RHEL) 78 based worker nodes worker01 and worker02 to the local IBM Spectrum Scale 5.1.0.0 storage cluster as IBM Spectrum Scale client nodes, we use the IBM Spectrum Scale installation toolkit and run the following commands (if the cluster was installed by using the installation toolkit):

```
# ./spectrumscale node add worker01.ocp4.scale.ibm.com
# ./spectrumscale node add worker02.ocp4.scale.ibm.com
# ./spectrumscale install [--precheck]
# ./spectrumscale deploy [--precheck]
```

For more information about using the installation toolkit, see [Installing IBM Spectrum Scale on Linux nodes with the installation toolkit](#) and [Adding nodes, NSDs, or file systems to an existing installation](#).

The two OpenShift worker nodes now show up as new clients in the IBM Spectrum Scale storage cluster:

```
# mmisccluster
```

GPFS cluster information
=====

```
GPFS cluster name:      SpectrumScale.ocp4.scale.ibm.com
GPFS cluster id:        16217308676025981087
GPFS UID domain:        SpectrumScale.ocp4.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	scale00.ocp4.scale.ibm.com	192.168.1.30	scale00.ocp4.scale.ibm.com	quorum-manager-perfmon
2	worker01.ocp4.scale.ibm.com	192.168.1.15	worker01.ocp4.scale.ibm.com	perfmon
3	worker02.ocp4.scale.ibm.com	192.168.1.16	worker02.ocp4.scale.ibm.com	perfmon

This configuration is a minimal configuration with a single quorum node that also hosts the IBM Spectrum Scale GUI and the two OpenShift worker nodes as clients. This configuration is not recommended for production environments. Typically, a customer has extra nodes for special roles (for example, protocol nodes and GUI nodes) and designations (manager and quorum). Special considerations must be made for the number of quorum nodes in the cluster (see [Quorum](#)). A node that is running the IBM Spectrum Scale GUI is required in the local and the remote IBM Spectrum Scale clusters for the IBM Spectrum Scale CSI Driver to communicate through the REST interface.

The local IBM Spectrum Scale cluster remotely mounts the IBM Spectrum Scale file systems that are named *ess3000_4M* and *ess3000_1M* from an IBM ESS 3000 storage system that provides the high-performance smart storage tier for the DAAA workflow. For more information about managing access to a remote IBM Spectrum Scale file system, see [Accessing a remote GPFS file system](#) and [Mounting a remote GPFS file system](#).

The remotely mounted file systems are configured as follows in the local IBM Spectrum Scale storage cluster:

```
# mmremoteclass show
Cluster name:      ess3000.bda.scale.ibm.com
Contact nodes:
fscs-fab3-3-a-priv.bda.scale.ibm.com,fscs-fab3-3-b-priv.bda.scale.ibm.com
SHA digest:        9c58d9df69804393571044a9f08b005db36807e4cb87637cad83870a9f74c24d
File systems:      ess3000_1M (ess3000_1M)  ess3000_4M (ess3000_4M)

# mmremotefs show
Local Name  Remote Name  Cluster name          Mount Point          Mount Options
ess3000_1M  ess3000_1M   ess3000.bda.scale.ibm.com /gpfs/ess3000_1M    rw
ess3000_4M  ess3000_4M   ess3000.bda.scale.ibm.com /gpfs/ess3000_4M    rw
```

The file system *ess3000_4M* is used for the DAAA workflow to provide the high-performance smart storage tier for data that is retrieved from warm and cold storage tiers like NFS, object, and IBM Spectrum Archive Enterprise Edition. The *ess3000_1M* file system is used as back-end storage for dynamically provisioned volumes through storage classes in OpenShift.

In addition to the 1 Gbps admin network, we add the **subnets 10.10.1.0** parameter to the Spectrum Scale cluster configuration command (**mmchconfig**) so that the IBM Spectrum Scale daemon can use the 100 Gbps high-speed network for data transfers. For more information about the network configuration and daemon communication, see [Using public and private IP addresses for GPFS nodes](#).

IBM Spectrum Scale CSI Driver

The IBM Spectrum Scale CSI Driver enables the provisioning of persistent volumes (PVs) with IBM Spectrum Scale as the storage back end for containerized workloads in OpenShift and Kubernetes.

In OpenShift, the IBM Spectrum Scale CSI Driver can be installed by using the IBM Spectrum Scale CSI operator, which can deploy and manage the CSI plug-in for IBM Spectrum Scale. The IBM Spectrum Scale CSI operator is available in the OperatorHub in the Red Hat OpenShift web console, as shown in Figure 5-2.

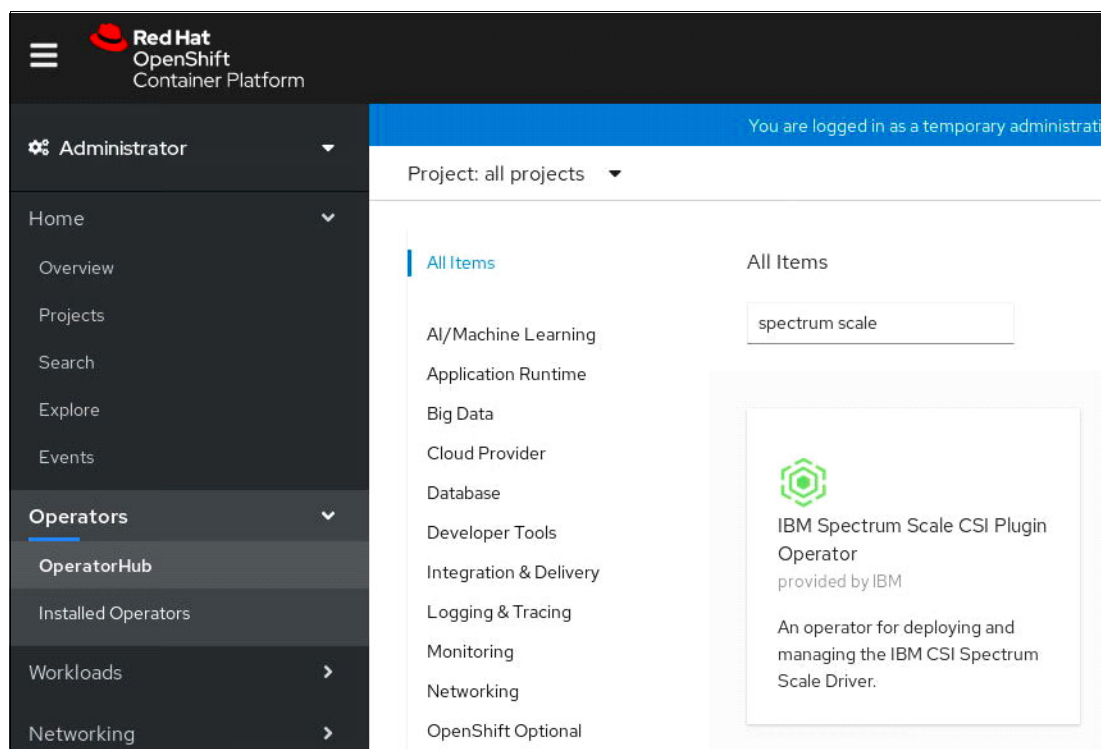


Figure 5-2 Installing the IBM Spectrum Scale CSI operator from the OpenShift OperatorHub window

The deployment of the IBM Spectrum Scale CSI plug-in requires a running IBM Spectrum Scale storage cluster with access to the IBM Spectrum Scale GUI (local and remote). In our PoC environment, we use a Red Hat OpenShift V4.5.9 compute cluster with IBM Spectrum Scale CSI Driver V2.0.0.

For more information and the full documentation of IBM Spectrum Scale CSI, see [IBM Spectrum Scale Container Storage Interface Driver 2.0.0](#) or [IBM Spectrum Scale CSI Driver for Container Persistent Storage](#), REDP-5589.

Follow the instructions at [Performing pre-installation tasks](#) to prepare your environment.

For OpenShift, these pre-installation steps are as follows:

1. Create an IBM Spectrum Scale user group “CsiAdmin” in the IBM Spectrum Scale GUI on the local and remote storage cluster (the IBM ESS 3000 storage cluster) if it does not exist:

```
# /usr/lpp/mmfs/gui/cli/mkusergrp CsiAdmin --role csiadmin
```

2. Create an IBM Spectrum Scale user in the “CsiAdmin” group in the IBM Spectrum Scale GUI on the local and remote storage cluster (the IBM ESS 3000 storage cluster):

```
#/usr/lpp/mmfs/gui/cli/mkuser <username> -p <password> -g CsiAdmin
```

3. Test access to the IBM Spectrum Scale GUI from the OpenShift worker nodes for IBM Spectrum Scale CSI:

```
# curl --insecure -u 'username:password' -X GET \
```

- ```

https://scale00.ocp4.scale.ibm.com:443/scalemgmt/v2/cluster
curl --insecure -u 'username:password' -X GET \
https://node12.bda.scale.ibm.com:443/scalemgmt/v2/cluster

```
4. Ensure that `perfileset-quota` on the file systems that are used by IBM Spectrum Scale CSI is set to No:

```
mmlsfs <file system name> --perfileset-quota
```
  5. Enable quota for all the file systems that are used for file set-based dynamic provisioning of PVs:

```
mmchfs <file system name> -Q yes
```
  6. Enable quota for the root user:

```
mmchconfig enforceFilesetQuotaOnRoot=yes -i
```
  7. Ensure that the `controlSetxattrImmutableSELinux` configuration parameter is set to yes:

```
mmchconfig controlSetxattrImmutableSELinux=yes -i
```
  8. To display the correct volume sizes in the containers, enable `filesetdf` on the file systems:

```
mmchfs <file system name> --filesetdf
```

Before installing the IBM Spectrum Scale CSI operator from the OperatorHub in the OpenShift web console, as shown in Figure 5-2 on page 54, the system admin must complete the following steps:

1. Create a namespace (also known as a *project* in OpenShift) for the IBM Spectrum Scale CSI Driver:

```
oc create namespace ibm-spectrum-scale-csi-driver
```
2. Create the secrets to hold the *username* and *password* for the IBM Spectrum Scale GUI:

```
oc create secret generic csi-local --from-literal=username=<username> \
--from-literal=password='<password>' -n ibm-spectrum-scale-csi-driver
oc create secret generic csi-remote --from-literal=username=<username> \
--from-literal=password='<password>' -n ibm-spectrum-scale-csi-driver
```

Here, the name of the secret `csi-local` refers to the *local* IBM Spectrum Scale storage cluster and `csi-remote` refers to the *remote* IBM ESS 3000 storage cluster that provides the high-performance smart storage tier that is remotely mounted on the local cluster.

We also label the GPU-accelerated (and RHEL78 based) worker nodes to specify where the IBM Spectrum Scale client is installed and the IBM Spectrum Scale CSI Driver should run:

```
oc label node worker01.ocp4.scale.ibm.com scale=true --overwrite=true
oc label node worker02.ocp4.scale.ibm.com scale=true --overwrite=true
```

```
oc get nodes -l scale=true
```

| NAME                        | STATUS | ROLES  | AGE   | VERSION         |
|-----------------------------|--------|--------|-------|-----------------|
| worker01.ocp4.scale.ibm.com | Ready  | worker | 4d21h | v1.18.3+47c0e71 |
| worker02.ocp4.scale.ibm.com | Ready  | worker | 4d21h | v1.18.3+47c0e71 |

Then, we install the IBM Spectrum Scale CSI operator from the OperatorHub in the OpenShift web console with the following custom resource YAML for the setup of the PoC in this page:

```

apiVersion: csi.ibm.com/v1
kind: CSIScaleOperator
metadata:
 name: ibm-spectrum-scale-csi
 labels:

```

```

 release: ibm-spectrum-scale-csi-operator
 app.kubernetes.io/name: ibm-spectrum-scale-csi-operator
 app.kubernetes.io/instance: ibm-spectrum-scale-csi-operator
 app.kubernetes.io/managed-by: ibm-spectrum-scale-csi-operator
 namespace: ibm-spectrum-scale-csi-driver
spec:
 provisionerNodeSelector:
 - key: scale
 value: 'true'
 attacherNodeSelector:
 - key: scale
 value: 'true'
 pluginNodeSelector:
 - key: scale
 value: 'true'
 scaleHostpath: "/gpfs/fs0"
 clusters:
 - secrets: csi-local
 restApi:
 - guiHost: scale00.ocp4.scale.ibm.com
 secureSslMode: false
 primary:
 primaryFs: fs0
 id: "16217308676025981087"
 - secrets: csi-remote
 restApi:
 - guiHost: node12.bda.scale.ibm.com
 secureSslMode: false
 id: "215057217487177715"
status: {}

```

Our PoC configuration is composed of the following components:

- ▶ A local IBM Spectrum Scale storage cluster with cluster ID 16217308676025981087 (found by running **mm1scluster**) and the IBM Spectrum Scale GUI running on node scale00.ocp4.scale.ibm.com (IP 192.168.1.30) with a local primary file system fs0 for hosting IBM Spectrum Scale CSI configuration data.
- ▶ A remote IBM ESS 3000 storage cluster with cluster ID 215057217487177715 (found by running **mm1scluster**) and the IBM Spectrum Scale GUI running on node node12.bda.scale.ibm.com (IP 192.168.1.52) hosting the remotely mounted file systems ess3000\_4M (for the high-performance smart storage tier) and ess3000\_1M (for the dynamic provisioning of persistent storage in OpenShift).

For more information about the IBM Spectrum Scale CSI Operator custom resource YAML, see [Operator](#) and [Remote cluster support](#).

### 5.3.3 Compute cluster

This section provides considerations and best practices for implementing the DAAA workflow on the compute cluster. All nodes in the compute cluster that are running workloads on data in IBM Spectrum Scale (the high-performance smart storage tier) must be added as IBM Spectrum Scale client nodes to the local IBM Spectrum Scale storage cluster (for more information, see 5.3.2, “IBM Spectrum Scale storage cluster” on page 51).

The task at hand is that a user wants to run a workload on a selected set of data that is prefetched and made available in the high-performance smart storage tier at the user's request. DAAA enables users to easily retrieve the data of interest no matter where the data is (no knowledge of the actual storage location is required). Furthermore, the requested data is made available (cached) in the high-performance smart storage tier close to the compute nodes that are accelerating data-intensive workloads like deep neural network (DNN) training workloads. When the user's job finishes, the prefetched data is automatically evicted from the high-performance smart storage tier to provide space for the next jobs, which eliminates unnecessary data copies and the associated data management efforts.

The personas that are involved are the system *admin*, who must provide and configure the necessary resources for the DAAA workflow, and the *user* who requests the data and eventually runs the workload on the compute cluster.

A user typically defines a workload as a queued batch job that is run in the user's context on the compute nodes in a scheduler like, for example, LSF or Slurm. Scripts and templates in the scheduler take care of running the job and providing the required resources.

As described in 4.4.2, “Compute cluster” on page 34, the user context and access to the data that is stored in the high-performance smart storage tier (the IBM Spectrum Scale storage cluster) requires different considerations in a traditional (HPC-like) compute cluster and in an orchestrated container platform like OpenShift and Kubernetes. In a traditional compute cluster, a job has access to the IBM Spectrum Scale file system that is mounted on the compute nodes in the regular context of the user (user ID and group ID) that runs the job.

However, in OpenShift or Kubernetes, the storage layer is abstracted from the user and access to the data in IBM Spectrum Scale requires extra steps like creating the PVs (which provide access to the requested data from the DAAA workflow), *persistent volume claims* (PVCs), and finally the Pods to run the workload. Here, the tasks are divided into tasks for the *system admin*, who must create the PVs that provide access to the prefetched data in the DAAA pipeline on demand, and the *user*, who must request the PVs through the PVCs to bind the PVs to their namespace and mount these PVs into the Pod that runs the workload. The correct matching of the PV (statically provisioned by the system admin) to the user's PVC with the requested data in IBM Spectrum Scale from the DAAA pipeline is an extra challenge that we must address.

In the following sections, we propose some best practices about how to deploy the DAAA pipeline on an OpenShift compute cluster with IBM Spectrum Scale by using the IBM Spectrum Scale CSI Driver to manage data access in IBM Spectrum Scale through PVs.

## The DAAA workflow on OpenShift with IBM Spectrum Scale CSI

The IBM Spectrum Scale CSI Driver can manage and provision PVs in Kubernetes and Red Hat OpenShift with IBM Spectrum Scale as the storage back end. The IBM Spectrum Scale CSI Driver is installed in OpenShift by using the IBM Spectrum Scale CSI operator, as described in “IBM Spectrum Scale CSI Driver” on page 53.

With the IBM Spectrum Scale CSI Driver, PVs can be *provisioned dynamically* on a user request (in a self-service manner by using *storage classes*) or statically by a system admin by using a directory path in IBM Spectrum Scale. Although dynamic provisioning provides a newly created (and empty) volume to a user, static provisioning also can provide access to existing data in IBM Spectrum Scale (for example, access to huge amounts of training data that is shared among all the data scientists in a team).

Furthermore, as a clustered parallel file system, IBM Spectrum Scale can share access to data in IBM Spectrum Scale across Pods and physical worker node boundaries in a *Read-Write-Many* (RWX) fashion so that containerized workloads can seamlessly scale out horizontally and efficiently use of as many of the available compute and GPU resources in the entire cluster as needed.

In the PoC that is described in this paper, we use *dynamic provisioning* through *storage classes* to provide the *individual workspace* for a user, where the user, for example, the data scientist, can persistently store the code, scripts, and models that they develop to run the artificial intelligence (AI) workloads. We use *static provisioning* to enable access to data in IBM Spectrum Scale that was specifically requested by the user through the DAAA pipeline and preinstalled into the high-performance smart storage tier. For more information about the configuration of IBM Spectrum Scale CSI and the various types of storage provisioning, see [Using IBM Spectrum Scale Container Storage Interface Driver](#).

In our PoC, we have a user “dean” who works in the namespace “dean” in an OpenShift environment, who plans to run a large DNN training workload on data that is returned by the DAAA pipeline. The user has a PV that is used as the user’s individual workspace for the code and model development, which is mounted at the mount point /workspace into the container of the user’s Pod. Then, the user requests specific data to be fetched through the DAAA pipeline to run a scheduled training job on it. This data is preinstalled into the high-performance smart storage tier and made accessible to the user through another set of PVs.

In our PoC example, the DAAA pipeline returns data from different storage architectures and systems, for example, NFS storage, object storage and IBM Spectrum Archive. The data is made available through different PVs and mounted into the container at mount points /data/cos/, /data/nfs/, and /data/arc/, as shown in Figure 5-3.

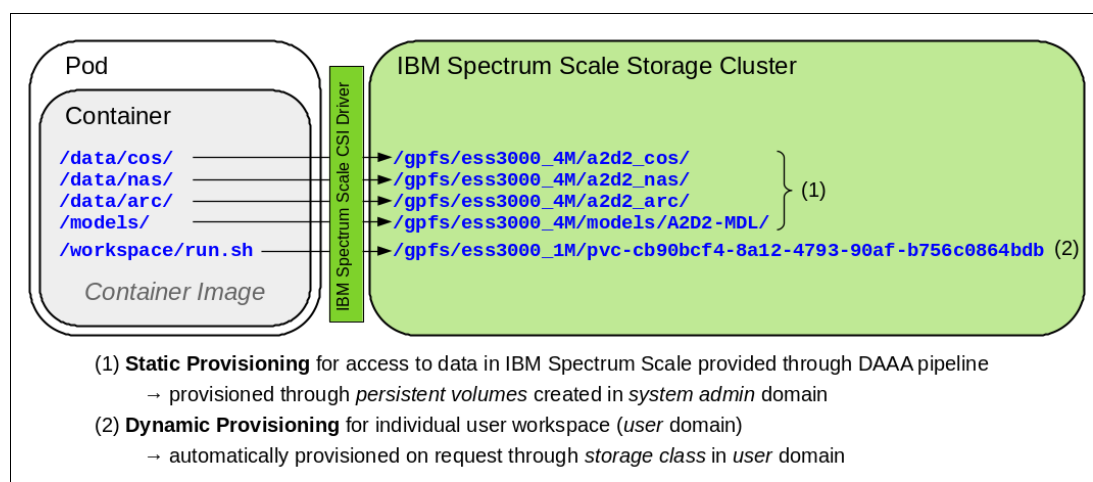


Figure 5-3 Mapping between mount points in OpenShift Pod and the IBM Spectrum Scale storage cluster

In addition, we have a file that is called content.txt in each of these mount points with a list of the prefetched files and their total path in that directory to be used for the batch job. All objects and files belonging to the original data location are visible under each mount point, but only the listed files and objects were retrieved and preinstalled into the high-performance smart storage tier.

We also make an extra Models bucket from the IBM Cloud Object Storage that is available in the container through another PV at the mount point `/models`. So, the newly trained *model* can be written back to the object store after the DAAA job finishes. This bucket in the object store is intended to hold all calculated *models* for a specific AI task so that these models can be easily shared among data scientists or even automatically picked up by an automated model test, verification, and deployment pipeline (a machine learning (ML) pipeline). This bucket also demonstrates that new data that is written to the AFM file sets in the high-performance smart storage tier is synchronized back to the object store, where it can easily be deployed.

Then, a scheduled DAAA batch job runs the user's code to run the AI training workload from the user's PV that is mounted under `/workspace` (the `run.sh` script) to process the requested data under `/data`, which was prefetched by the DAAA pipeline and preinstalled into the high-performance smart storage tier.

After a successful run of the batch job, the results (the *model*) are stored in the user's individual PV (`/workspace`) and copied to the `/models` directory. The preinstalled data is evicted from the high-performance smart storage tier, and the temporarily created set of PVs granting access to the data (`/data`) and Models bucket (`/models`) is deleted.

Alternatively, a user could also interactively work with the DAAA workflow. For example, in Jupyter Notebooks, the user can prefetch and preinstall the requested data into the high-performance smart storage tier and make the data accessible to the user's namespace through a set of PVs and PVCs. The data remains accessible until the user decides to delete the PVCs, which triggers the eviction of the data from the high-performance smart storage tier.

### ***Dynamic provisioning for an individual workspace***

The user's code development and deployment happens in Pods that are running in the user's namespace.

The user “dean” requests a new (empty) volume through a PVC that is used as an individual workspace in the user's Pods:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: "dean-workspace-pvc"
spec:
 storageClassName: remotefs-1m
 accessModes:
 - ReadWriteMany
 resources:
 requests:
 storage: 100Gi
```

Here, the user requests a volume of 100 GiB to be automatically created from a *storage class* that is named `remotefs-1m` without further involvement of the system admin.

After the claim is successfully issued, a new PV is created and bound to the user's namespace. Then, the user can mount the volume into one or more Pods:

```
apiVersion: v1
kind: Pod
metadata:
 name: daaa-pod
spec:
 containers:
```

```

- name: daaa-pod
 image: [...]
 volumeMounts:
 - name: vol5
 mountPath: "/workspace"
 volumes:
 - name: vol5
 persistentVolumeClaim:
 claimName: dean-workspace-pvc

```

Here, the PV that is bound to the PVC `dean-workspace-pvc` is mounted into the Pod's container under the mount point `/workspace`, which provides individual persistent storage to the user to develop and store code and models that later will be used to run the training workload of the DNN on the data that is requested from the DAAA pipeline.

The system admin had to prepare only the *storage class* in advance, as follows:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: remotefs-1m
provisioner: spectrumscale.csi.ibm.com
parameters:
 volBackendFs: "ess3000_1M"
 clusterId: "215057217487177715"
reclaimPolicy: Delete

```

The `clusterId` is the cluster ID of the remote cluster (the one that owns the file system), and `volBackendFs` with `ess3000_1M` specifies the file system that is used as the back end to store the user data. For more information about storage classes and the available options, see [Storage class](#).

### ***Static provisioning for accessing data that is provided by the DAAA pipeline***

The data that is prefetched and preinstalled into the high-performance smart storage tier through the DAAA workflow must be made accessible to a user in OpenShift through PVs that are created in the *system admin* domain.

For each of the three storage architectures in our PoC (IBM Cloud Object Storage, NFS, and IBM Spectrum Archive), we have a related file set that is linked to the remotely mounted IBM Spectrum Scale file system on the local IBM Spectrum Scale storage cluster. To provide access to the data in such a file set, the system admin creates a PV, for example:

```

apiVersion: v1
kind: PersistentVolume
metadata:
 name: daaa-cos-pv01
 labels:
 type: cos
 data: a2d2
spec:
 storageClassName: static
 capacity:
 storage: 100Gi
 accessModes:
 - ReadWriteMany
 csi:
 driver: spectrumscale.csi.ibm.com

```



```
volumeHandle:
"16217308676025981087;099B6A7A:5EB99743;path=/gpfs/ess3000_4M/a2d2_cos"
```

The PV manifest requires a `volumeHandle` with the cluster ID of the local storage cluster (16217308676025981087, as shown by running `mm1scluster`), the file system UID (099B6A7A:5EB99743, as shown by running `mm1sfs [file system] --uid`), and the local mount path to the data (`path=/gpfs/ess3000_4M/a2d2_cos`), and the `accessModes` that are supported (here `RWX`) and the capacity (here 100 GiB).

In addition, we add the `storageClassName: static` as an annotation<sup>1</sup> and use Kubernetes *labels* to allow a specific match between a PVC and this specific PV based on the data it represents. Otherwise, the match is based only on the requested *access mode* and *capacity*. Labels are key value pairs, and more labels (for example, a unique ID or a username) can be used to allow more specific matches with a higher granularity.

This volume can now be requested by a user and bound to the user's namespace through a PVC as follows:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: "daaa-cos-pvc"
spec:
 storageClassName: static
 accessModes:
 - ReadWriteMany
 resources:
 requests:
 storage: 100Gi
 selector:
 matchLabels:
 type: cos
 data: a2d2
```

Using the storage class annotation `static` and the labels `type: cos` and `data: a2d2` ensures that the PVC is matched to the correct PV, which has the same labels. Now, the user can request a specific PV based on the data that it holds. The storage class annotation and labels take precedence over the access mode and capacity.

In this paper, we use Kubernetes labels and a storage class annotation to provide an extended concept of static provisioning that goes beyond the information that is described in [Static provisioning](#).

A single PV can be bound only to one PVC, and cannot be bound to multiple PVCs in different namespaces. If the same PV is needed in different namespaces, then multiple PVs with the same manifest but different names (`daaa-cos-pv02`, `daaa-cos-pv03`, and so on) can be created.

---

<sup>1</sup> This is an annotation. It is not a real storage class like we use with dynamic provisioning. This annotation ensures that a volume is created from the default storage class and not from the pool of pre-provisioned PVs in case such a default storage class is defined, and no storage class is specified in the PVC.

### ***Running the AI workload as a job in OpenShift by using a Helm chart***

In the DAAA workflow that is used in this PoC, we run the deployment of the user's AI batch job in the context of the OpenShift system admin because the creation of the PV requires the system admin context. However, the PVC and the Pod that is running the *workload* is run in the user's namespace as a Kubernetes job to ensure that the AI workload is run in a user context with an arbitrarily determined user ID by OpenShift. The workload is not run in a system admin or even a privileged context.

We created similar PVs and PVCs for all three storage categories, for example, daaa-cos-pvc, daaa-nfs-pvc, and daaa-arc-pvc, plus the *Models* bucket daaa-models-pvc, so we can run the Kubernetes job in the user's namespace as follows:

```
apiVersion: batch/v1
kind: Job
metadata:
 name: daaa-job
spec:
 template:
 spec:
 containers:
 - name: daaa-job
 image: [container-image]:[container-tag]
 imagePullPolicy: IfNotPresent
 command: ["/bin/sh"]
 args: ["/workspace/run.sh"]
 volumeMounts:
 - name: vol1
 mountPath: "/data/cos"
 readOnly: true
 - name: vol2
 mountPath: "/data/nfs"
 readOnly: true
 - name: vol3
 mountPath: "/data/arc"
 readOnly: true
 - name: vol4
 mountPath: "/models"
 - name: vol5
 mountPath: "/workspace"
 volumes:
 - name: vol1
 persistentVolumeClaim:
 claimName: daaa-cos-pvc
 - name: vol2
 persistentVolumeClaim:
 claimName: daaa-nfs-pvc
 - name: vol3
 persistentVolumeClaim:
 claimName: daaa-arc-pvc
 - name: vol4
 persistentVolumeClaim:
 claimName: daaa-models-pvc
 - name: vol5
 persistentVolumeClaim:
 claimName: dean-workspace-pvc
```

```
restartPolicy: Never
backoffLimit: 3
```

The three volumes holding the data that was prefetched from the DAAA pipeline (/data/...), the volume of the Models bucket on IBM Cloud Object Storage to export the newly trained model (/models), and the user's private volume holding the user's executable code (/workspace) are mounted into the container and started as Kubernetes job by running the **/workspace/run.sh** script.

To easily compose the required PV, PVC, and Pod manifests (YAML) for a scheduled DAAA job, we use a *Helm* chart in this PoC instead of deploying and creating individual YAML manifests for every job. A Helm chart conveniently bundles all the required YAML files in a single package based on YAML templates for the individual components and default values.

Then, this Helm chart can be deployed in one step in OpenShift with different parameters for each DAAA job and with the individual YAML manifests adjusted and applied, for example, including the PVC name (OC\_PVC) of the user's individual workspace with the executable code, the path to the executable code (OC\_PATH\_EXEC), and a unique identifier (JOBID) for each deployment of these resources in the user's namespace (OC\_NAMESPACE) in OpenShift.

Here is an example of installing and removing such a deployment for a DAAA job:

```
helm install daaa-$JOBID helm/daaa \
 --set user.pvc="$OC_PVC" \
 --set user.command="$OC_PATH_EXEC" \
 -n $OC_NAMESPACE

helm delete daaa-$JOBID -n $OC_NAMESPACE
```

The Helm chart comprises the following files:

```
helm/daaa:
Chart.yaml
LICENSE
templates/
values.yaml

helm/daaa/templates:
daaa-arc-pv.yaml
daaa-arc-pvc.yaml
daaa-cos-pv.yaml
daaa-cos-pvc.yaml
daaa-job.yaml
daaa-models-pvc.yaml
daaa-models-pv.yaml
daaa-nfs-pv.yaml
daaa-nfs-pvc.yaml
_helpers.tpl
NOTES.txt
```

The `values.yaml` file introduces the available variables with their default values. All the YAML files under `templates` represent the individual templates for the YAML manifests of the required resources for the OpenShift deployment of the DAAA job.

We run the Helm chart as user “dean” on the LSF node, so we must make sure that `helm` is installed on the LSF node and that the appropriate `.kube/config` file with the proper credentials and cluster context is provided (here we used the `system:admin` context because the job must create PVs, which requires `system:admin` privileges). For more information about Helm, see [Installing Helm](#).

The variables and default values in the Helm chart that is used in our PoC implementation are described in the `values.yaml` file:

```
Spectrum Scale Client File System and AFM Fileset Directories
GPFSlocal:
 clusterID: "16217308676025981087"
 fileSystemID: "099B6A7A:5EB99743"
 mainPath: "/gpfs/ess3000_4M"
 arcDir: "a2d2_arc"
 nfsDir: "a2d2_nfs"
 cosDir: "a2d2_cos"
 modDir: "models/A2D2-MDL"

Container image to use for job/pod
image:
 name: alpine
 tag: 3.12
 pullPolicy: IfNotPresent

User private PVC with path to script for workload execution
(no uid/gid need to be defined; Job obtains arbitrary user id)
user:
 uid: ""
 gid: ""
 pvc: "dean-workspace-pvc"
 command: "run.sh"
```

Finally, the Helm chart running the Kubernetes job for the intended (here simulated) AI training workload in OpenShift is run through the `runAnalytics.sh` script in the DAAA workflow.

The `runAnalytics.sh` script performs the following basic steps for the deployment of the DAAA job in OpenShift:

1. Install the Helm chart in the user's namespace:

```
helm install "daaa-$JOBID" ${SCRIPT_PATH}/helm/daaa \
 --set user.pvc="$OC_PVC" \
 --set user.command="$OC_PATH_EXEC" \
 -n $OC_NAMESPACE
```
2. Wait for the Kubernetes job to complete:

```
oc wait --for=condition=complete --timeout=180s \
 "job.batch/daaa-$JOBID-job" -n $OC_NAMESPACE
```
3. Collect logs from the completed Kubernetes job:

```
oc logs job.batch/daaa-$JOBID-job -n $OC_NAMESPACE
```
4. Clean up and delete all created DAAA resources (JOB, PVCs, and PVs):

```
helm delete daaa-$JOBID -n $OC_NAMESPACE
```

The Helm chart to create the OpenShift resources (like PVs, PVCs, and the Kubernetes job to run the workload) and the `runAnalytics.sh` sample script that was used in this PoC can be found on [GitHub](#).

## 5.4 Data catalog layer

The following sections provide more information about DAAA deployment considerations for the components that are part of the data catalog layer.

### 5.4.1 IBM Spectrum Discover

The following paragraphs provide details about best practices when deploying IBM Spectrum Discover.

More details about the IBM Spectrum Discover configuration in the PoC environment are included in this section.

#### **Best practices**

The following sections provide best practices for deploying, enriching, and using the data catalog.

#### ***Deployment instructions***

The instructions for deploying IBM Spectrum Discover depend on the deployment model. If IBM Spectrum Discover is being deployed as a virtual appliance, follow the instructions at [IBM Knowledge Center](#).

If IBM Spectrum Discover is deployed as an application in an OpenShift Container Platform (Red Hat OCP) instance, follow the instructions at [IBM Knowledge Center](#).

#### ***Enriching the catalog***

Much of the power of IBM Spectrum Discover lies in collecting, deriving, organizing, and searching for metadata across a heterogeneous storage environment. Adding metadata to the catalog allows for more sophisticated and targeted queries, which means data scientists can more easily and accurately find the right data set to analyze.

There are several ways to enrich the IBM Spectrum Discover catalog:

- ▶ Applying custom metadata by using policies
- ▶ Deriving metadata by using content search policies
- ▶ Ingesting tags from a pre-curated source

*Tag ingest* is a new function of IBM Spectrum Discover that is built specifically for DAAA. In IBM Spectrum Discover V2.0.4, tag ingest is limited to COS sources. More data source types will be added in future releases. For instructions about how to ingest pre-curated tags or labels, see [IBM Knowledge Center](#).

#### ***Using the catalog***

The information that is stored in IBM Spectrum Discover can be searched either by using the GUI or a rich set of REST APIs.

### Using a workload manager

If a workload manager is being used in the AI pipeline, then the IBM Spectrum Discover “search” endpoint can be used to programmatically search for which files or objects to cache into the high-performance tier. For more information, see [IBM Knowledge Center](#).

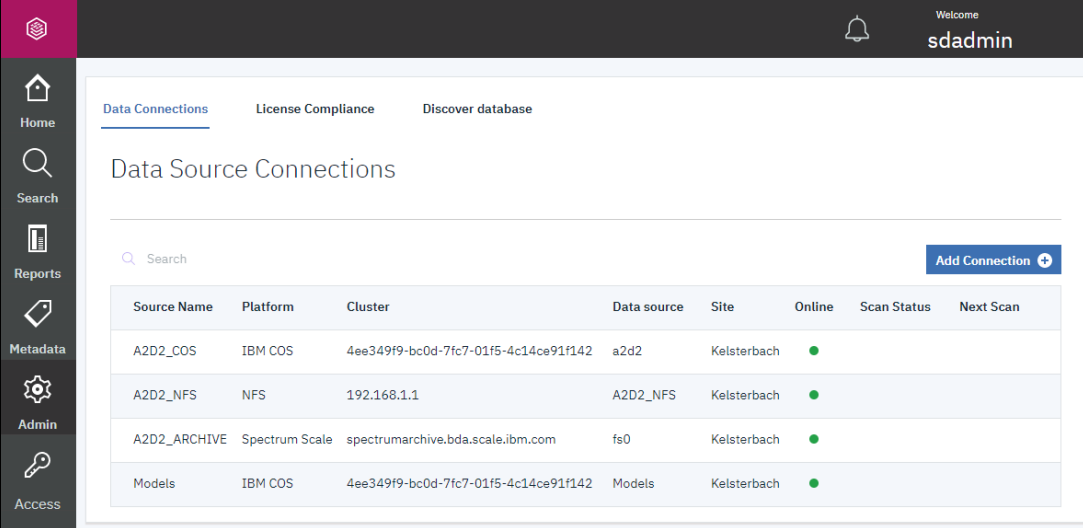
### Using the IBM Spectrum Discover UI

IBM Spectrum Discover supports both a visual query, where users can “shop” for their data, and an ad hoc query for more sophisticated searches. A data scientist can use either option to identify the data set to be used in the AI pipeline. The data set can be represented by either a filter that is based on a metadata query, or list of file and objects. This information can be used in a data movement policy that caches the data in the high-performance tier. For more information, see [IBM Knowledge Center](#).

### Configuring IBM Spectrum Discover in the proof of concept environment

In our PoC environment, we use an IBM Spectrum Discover installation as the metadata search engine. For details about how to configure an IBM Spectrum Discover system, see [IBM Knowledge Center](#). In the following section, only the configuration details for DAAA and our PoC are described.

For each capacity tier storage, we configure a data source connection, as shown in Figure 5-4.



| Source Name  | Platform       | Cluster                              | Data source | Site        | Online | Scan Status | Next Scan |
|--------------|----------------|--------------------------------------|-------------|-------------|--------|-------------|-----------|
| A2D2_COS     | IBM COS        | 4ee349f9-bc0d-7fc7-01f5-4c14ce91f142 | a2d2        | Kelsterbach | ●      |             |           |
| A2D2_NFS     | NFS            | 192.168.1.1                          | A2D2_NFS    | Kelsterbach | ●      |             |           |
| A2D2_ARCHIVE | Spectrum Scale | spectrumarchive.bda.scale.ibm.com    | fs0         | Kelsterbach | ●      |             |           |
| Models       | IBM COS        | 4ee349f9-bc0d-7fc7-01f5-4c14ce91f142 | Models      | Kelsterbach | ●      |             |           |

Figure 5-4 Showing the configured data source connections in the IBM Spectrum Discover GUI

The connections are configured as follows:

- NFS Server connection  
Connection Name: A2D2\_NFS  
Connection Type: Network File System  
Select a Collection: not selected  
Schedule Data Scan: not selected  
Datasource: A2D2\_NFS  
Export Path: /data2/nfs  
Host: 192.168.1.1  
site (Optional): Kelsterbach

► IBM Cloud Object Storage connection

Bucket: a2d2  
Connection Name: A2D2\_COS  
Connection Type: Cloud Object Storage  
Select a Collection: not selected  
Schedule Data Scan: not selected

To get help about how to retrieve the needed details, click the information icon:

Manager Api User: admin  
Manager Api Password: oooooooooo  
UUID: 4ee349f9-bc0d-7fc7-01f5-4c14ce91f142  
Manager Host: 192.168.1.210  
Vault: a2d2  
Accesser Host: 192.168.1.211  
Accesser Access Key: AsJIVCOLQqGuAGMcMnwR  
Accesser Secret Key: oooooooooo  
site (Optional): Kelsterbach

Bucket: Models  
Connection Name: Models  
Connection Type: Cloud Object Storage  
Select a Collection: not selected  
Schedule Data Scan: not selected

To get help about how to retrieve the needed details, click the information icon:

Manager Api User: admin  
Manager Api Password: oooooooooo  
UUID: 4ee349f9-bc0d-7fc7-01f5-4c14ce91f142  
Manager Host: 192.168.1.210  
Vault: Models  
Accesser Host: 192.168.1.211  
Accesser Access Key: AsJIVCOLQqGuAGMcMnwR  
Accesser Secret Key: oooooooooo  
site (Optional): Kelsterbach

► IBM Spectrum Scale connection

Connection Name: A2D2\_ARCHIVE  
Connection Type:  
Enable live events: selected  
Select a Collection: not selected  
Schedule Data Scan: not selected

To get help about how to retrieve the needed details, click the information icon:

User: root  
Password: oooooooooo  
Working Directory: /tmp/  
Scan Directory: /ibm/fs0/archive  
Site (Optional): Kelsterbach  
Cluster: spectrumarchive.bda.scale.ibm.com  
Host: 192.168.1.240  
Filesystem: fs0  
Node List: all

For IBM Cloud Object Storage, the notification service must be enabled and configured in the COS management GUI, as described in “Configuring NAS Filer in the proof of concept environment” on page 40. Enabling IBM Spectrum Scale to push metadata updates can be run by setting **Enable live events** during the IBM Spectrum Scale connection configuration in IBM Spectrum Discover. IBM Spectrum Discover remotely configures IBM Spectrum Scale by using its watch folders technology. Automatic metadata updates can be configured for IBM Cloud Object Storage and IBM Spectrum Scale and other products. For more information, see [IBM Knowledge Center](#).

Further regular scans can be scheduled in IBM Spectrum Discover. The IBM Spectrum Discover REST interface can request a scan. For sample code, see Appendix A, “Code samples” on page 71.

## 5.5 The Data Accelerator for AI and Analytics interface glue code

This section provides more details about our PoC and some explanations of the parts of the PoC by using sample scripts. For more information about these sample scripts, see Appendix A, “Code samples” on page 71.

The following sample is based on using IBM Spectrum LSF as the workload manager. If your environment does not have LSF, rework script **DAAA.sh** and remove the **bsub** and **-J** and **-w** options so that only the calls to the scripts stay. The called scripts are blocking scripts, which mean that they return after the subaction is done.

Place the sample scripts in a directory on the IBM Spectrum LSF host that is part of the *PATH* variable so that LSF can run it and set the executable flag. In our case, we placed it in `/usr/local/bin/DAAA` and provided 755 access to the root user and the group users (ensure that “dean” is part of the group users).

The **DAAA.sh** script is run by IBM Spectrum LSF and in the **AV.cmd** script that is provided by the job submission template. The gathered user input is provided to **DAAA.sh** as arguments.

The flow can also be started by a direct execution of the **DAAA.sh** script, such as when a new data ingest finishes or a user runs the whole flow from a Jupyter Notebook. A sample might look like the following string:

```
./DAAA.sh -oc_pvc= dean-workspace-pvc -oc_namespace=dean -oc_path_exec=run.sh
-min_bicycle=900 -max_bicycle=1000
```

Here are the steps of a typical workload flow, which is described in 3.1.2, “Analytic usage phase” on page 21:

1. The metadata details that are provided by the **DAAA.sh** script are reformatted into a query that is understood by IBM Spectrum Discover by the **searchDiscover.sh** script, which calls **discover\_search.sh**. This is an example of how to call the IBM Spectrum Discover REST interface. IBM Spectrum Discover replies with a list of files that matches the query, and provides the storage system and the path in which the data is located. The **searchDiscover.sh** script filters the data for the process.
2. In our PoC, we reused the existing commands to create AFM relationships, as described in “Active File Management to cloud object storage administration commands” on page 43. It is possible to call the commands on demand, which are followed by **mmunlinkfileset** and **mmde1fileset** after the workload finishes.



3. The **prepFileLists.sh** script provides an example of how to create the prefetched file lists per storage system. It also creates lists for files that must be recalled from tape. To recall from tape, use the **recallArchive.sh** script, which connects user “archi” to the IBM Spectrum Archive Enterprise Edition server and runs the command **eeadm recall <filelist>**. The file list that is returned by this command includes the fully qualified names of the files that must be recalled. The prefetch lists are provided to IBM Spectrum Scale AFM, which starts prefetching the data. The **cacheDataIn.sh** script calls a script on the AFM Gateway node that performs the AFM call and monitors the success of the prefetch. The script also can create a content list that can be used as input for the analytic workload scripts.
4. After the prefetch, or parts of it, is done, the workload starts. The **runAnalytics.sh** script provides a sample script that creates the storage classes, PV manifest files, PVCs, and PVs.
5. The **runAnalytics.sh** script provides an example of how to start and monitor Pods in the OpenShift environment that run our workload.
6. Cached data that is no longer needed is evicted by IBM Spectrum Scale AFM. The **evictDataOut.sh** script shows an example.
7. OpenShift resources that are no longer needed are deleted. The **runAnalytics.sh** script shows an example.
8. The **cleanup.sh** script ensures that temporary data is removed. Do not run this script to review details about, for example, the created prefetch lists.





## Code samples

The sample code that is used for our proof of concept (PoC) environment is open source and can be found at [GitHub](#).



# Related publications

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

- ▶ *Deployment and Usage Guide for Running AI Workloads on Red Hat OpenShift and NVIDIA DGX Systems with IBM Spectrum Scale*, REDP-5610
- ▶ *IBM Cloud Object Storage Concepts and Architecture System Edition*, REDP-5537
- ▶ *IBM Spectrum Scale CSI Driver for Container Persistent Storage*, REDP-5589
- ▶ *IBM Spectrum Archive Enterprise Edition V1.3.0.6: Installation and Configuration Guide*, SG24-8333

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](https://ibm.com/redbooks)

## Online resources

These websites are also relevant as further information sources:

- ▶ IBM Cloud Object Storage:
  - IBM Cloud Object Storage System - Main IBM Knowledge Center Page:  
<https://www.ibm.com/support/knowledgecenter/STXNRM>
  - CSO API (Amazon Simple Storage Service (Amazon S3) compatible):  
[https://www.ibm.com/support/knowledgecenter/STXNRM\\_3.15.1/coss.doc/csoApi\\_title.html](https://www.ibm.com/support/knowledgecenter/STXNRM_3.15.1/coss.doc/csoApi_title.html)
- ▶ IBM Spectrum Archive Enterprise Edition:
  - IBM Spectrum Archive Editions:  
<https://www.ibm.com/products/data-archive>
  - IBM Spectrum Archive Enterprise Edition at IBM Knowledge Center:  
[https://www.ibm.com/support/knowledgecenter/en/ST9MBR\\_1.3.0/ltfs\\_ee\\_ichome.html](https://www.ibm.com/support/knowledgecenter/en/ST9MBR_1.3.0/ltfs_ee_ichome.html)
  - IBM Spectrum Scale ILM Policy Guide for IBM Spectrum Archive Enterprise Edition:  
<https://www.ibm.com/support/pages/node/6260749>
  - SNIA LTFS Format Specification  
[http://www.snia.org/tech\\_activities/standards/curr\\_standards/ltfs](http://www.snia.org/tech_activities/standards/curr_standards/ltfs)

► Data set:

– Audi Autonomous Driving Dataset (A2D2) citation:

```
@article{geyer2020a2d2,title={{A2D2: Audi Autonomous Driving Dataset}},
author={Jakob Geyer and Yohannes Kassahun and Mentar Mahmudi and Xavier
Ricou and Rupesh Durgesh and Andrew S. Chung and Lorenz Hauswald and Viet
Hoang Pham and Maximilian M{\u}hllegg and Sebastian Dorn and Tiffany
Fernandez and Martin J{\\"a}nicke and Sudesh Mirashi and Chiragkumar Savani
and Martin Sturm and Oleksandr Vorobiov and Martin Oelker and Sebastian
Garreis and Peter Schuberth},
year={2020},
eprint={2004.06320},
archivePrefix={arXiv},
primaryClass={cs.CV},
url = {https://www.a2d2.audi}}
```

– *A2D2: Audi Autonomous Driving Dataset:*

<https://arxiv.org/pdf/2004.06320.pdf>

– Driving Dataset Downloads and Citation

<https://www.a2d2.audi/a2d2/en/download.html>

– Liability and Copyright (licensed material was not modified for this study):

<https://www.a2d2.audi/a2d2/en/legal.html>

– Public License:

<https://aev-autonomous-driving-dataset.s3.eu-central-1.amazonaws.com/LICENSE.txt>

This data set is released under the CC BY-ND 4.0 license:

<https://creativecommons.org/licenses/by-nd/4.0/>

## Help from IBM

IBM Support and downloads

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

IBM Global Services

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





REDP-5623-00

ISBN 0738459321

Printed in U.S.A.

Get connected

