100 XP

Introduction

1 minute

Many organizations have spent the last two decades building data warehouses and business intelligence (BI) solutions based on relational database systems. Many BI solutions have lost out on opportunities to store unstructured data due to cost and complexity in these types of data in databases.

Data lakes have become a common solution to this problem. A data lake provides file-based storage, usually in a distributed file system that supports high scalability for massive volumes of data. Organizations can store structured, semi-structured, and unstructured files in the data lake and then consume them from there in big data processing technologies, such as Apache Spark.

Azure Data Lake Storage Gen2 provides a cloud-based solution for data lake storage in Microsoft Azure, and underpins many large-scale analytics solutions built on Azure.

Next unit: Understand Azure Data Lake Storage Gen2

Continue

Azure Data Lake Storage combines a file system with a storage platform to help you quickly identify insights into your data. Data Lake Storage builds on Azure Blob storage capabilities to optimize it specifically for analytics workloads. This integration enables analytics performance, the tiering and data lifecycle management capabilities of Blob storage, and the high-availability, security, and durability capabilities of Azure Storage.

Benefits

Data Lake Storage is designed to deal with this variety and volume of data at exabyte scale while securely handling hundreds of gigabytes of throughput. With this, you can use Data Lake Storage Gen2 as the basis for both real-time and batch solutions.

Hadoop compatible access

A benefit of Data Lake Storage is that you can treat the data as if it's stored in a Hadoop Distributed File System (HDFS). With this feature, you can store the data in one place and access it through compute technologies including Azure Databricks, Azure HDInsight, and Azure Synapse Analytics without moving the data between environments. The data engineer also has the ability to use storage mechanisms such as the parquet format, which is highly compressed and performs well across multiple platforms using an internal columnar storage.

Security

Data Lake Storage supports access control lists (ACLs) and Portable Operating System Interface (POSIX) permissions that don't inherit the permissions of the parent directory. In fact, you can set permissions at a directory level or file level for the data stored within the data lake, providing a much more secure storage system. This security is configurable through technologies such as Hive and Spark or utilities such as Azure Storage Explorer, which runs on Windows, macOS, and Linux. All data that is stored is encrypted at rest by using either Microsoft or customer-managed keys.

Microsoft or customer-managed keys.

Performance

Azure Data Lake Storage organizes the stored data into a hierarchy of directories and subdirectories, much like a file system, for easier navigation. As a result, data processing requires less computational resources, reducing both the time and cost.

Data redundancy

Data Lake Storage takes advantage of the Azure Blob replication models that provide data redundancy in a single data center with locally redundant storage (LRS), or to a secondary region by using the Geo-redundant storage (GRS) option. This feature ensures that your data is always available and protected if catastrophe strikes.

> **Tip**
>
> Whenever planning for a data lake, a data engineer should give thoughtful consideration to structure, data governance, and security. This should include consideration of factors that can influence lake structure and organization, such as:
>
> - Types of data to be stored
> - How the data will be transformed
> - Who should access the data
> - What are the typical access patterns
>
> This approach will help determine how to plan for access control governance across your lake. Data engineers should be proactive in ensuring that the lake doesn't become the proverbial data swamp which becomes inaccessible and non-useful to users due to the lack of data governance and data quality measures. Establishing a baseline and following best practices for Azure Data Lake will help ensure a proper and robust implementation that will allow the organization to grow and gain insight to achieve more.

Next unit: Enable Azure Data Lake Storage Gen2 in Azure Storage

Continue >

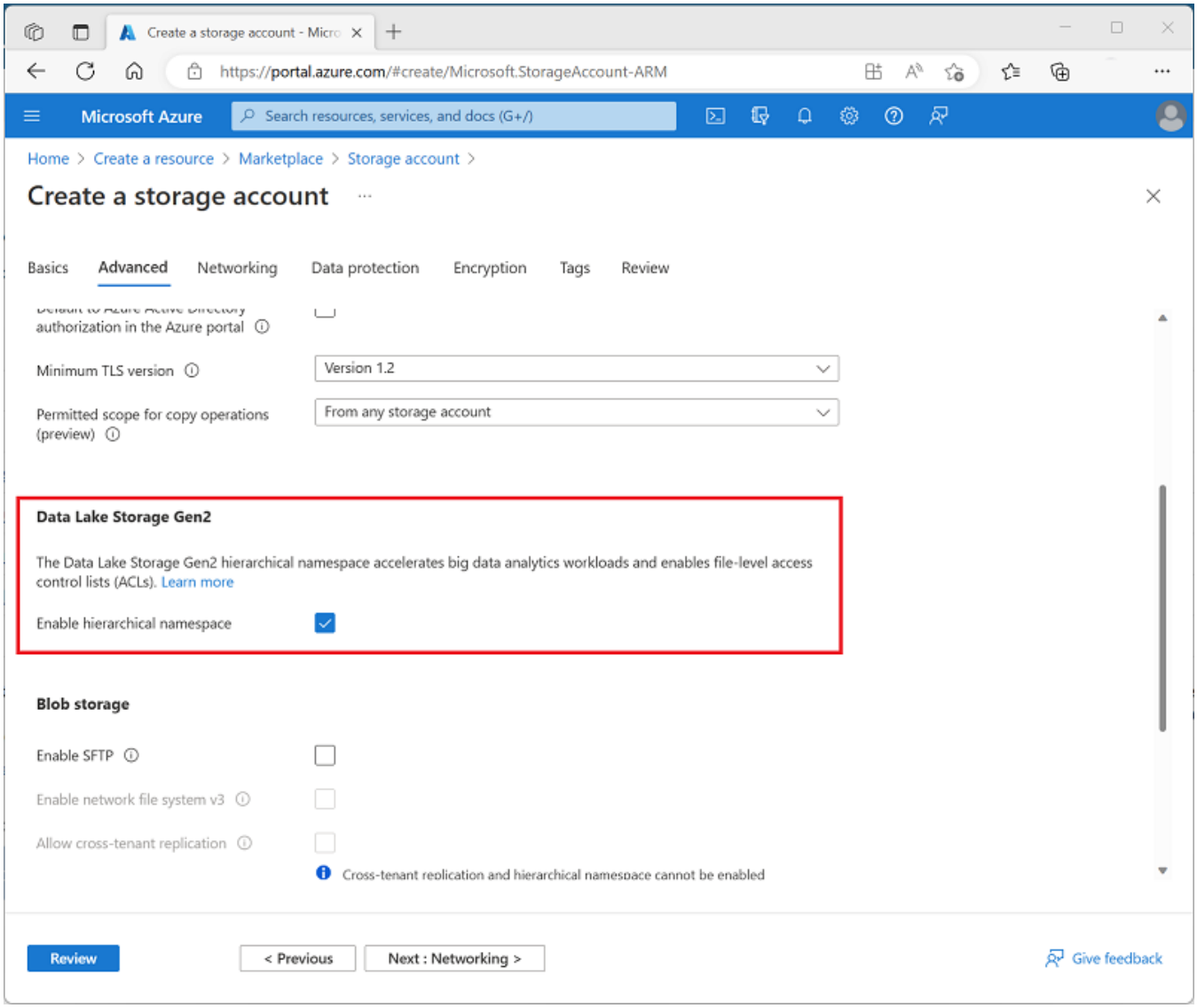
100 XP

Enable Azure Data Lake Storage Gen2 in Azure Storage

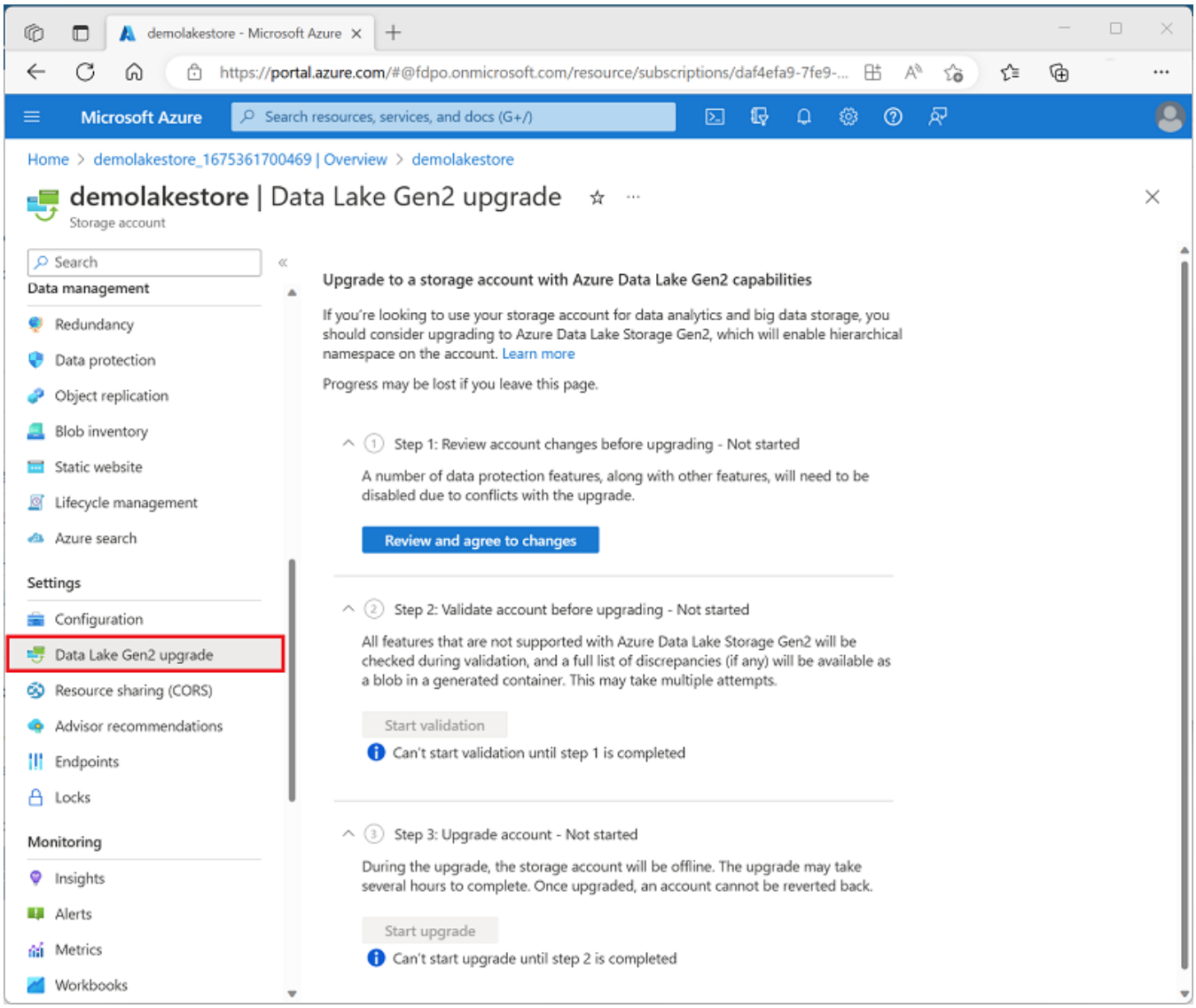
5 minutes

Azure Data Lake Storage Gen2 isn't a standalone Azure service, but rather a configurable capability of a **StorageV2 (General Purpose V2)** Azure Storage.

To enable Azure Data Lake Storage Gen2 in an Azure Storage account, you can select the option to **Enable hierarchical namespace** in the **Advanced** page when creating the storage account in the Azure portal:



Alternatively, if you already have an Azure Storage account and want to enable the Azure data Lake Storage Gen2 capability, you can use the **Data Lake Gen2 upgrade** wizard in the Azure portal page for your storage account resource.



Next unit: Compare Azure Data Lake Store to Azure Blob storage

Continue >

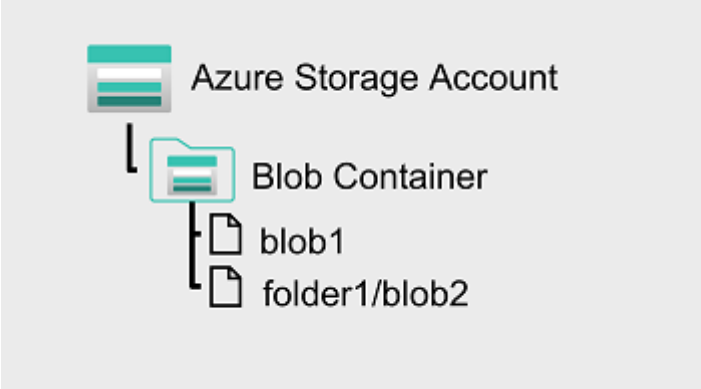
How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Compare Azure Data Lake Store to Azure Blob storage

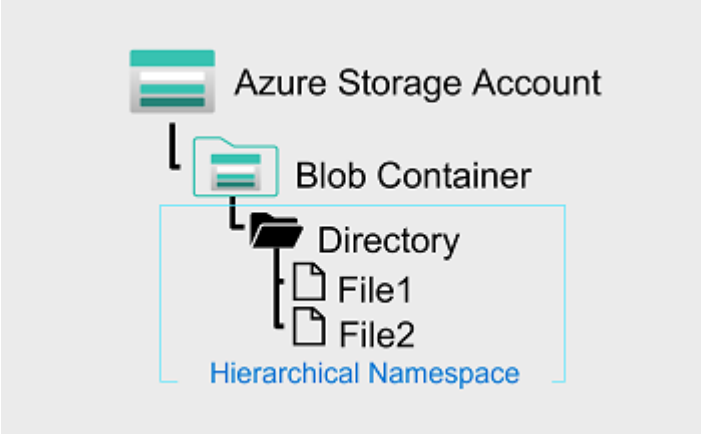
5 minutes

In Azure Blob storage, you can store large amounts of unstructured ("object") data in a flat namespace within a blob container. Blob names can include "/" characters to organize blobs into virtual "folders", but in terms of blob manageability the blobs are stored as a single-level hierarchy in a flat namespace.



You can access this data by using HTTP or HTTPs

Azure Data Lake Storage Gen2 builds on blob storage and optimizes I/O of high-volume data by using a hierarchical namespace that organizes blob data into *directories*, and stores metadata about each directory and the files within it. This structure allows operations, such as directory renames and deletes, to be performed in a single atomic operation. Flat namespaces, by contrast, require several operations proportionate to the number of objects in the structure. Hierarchical namespaces keep the data organized, which yields better storage and retrieval performance for an analytical use case and lowers the cost of analysis.



Tip

If you want to store data *without performing analysis on the data*, set the **Hierarchical Namespace** option to **Disabled** to set up the storage account as an Azure Blob storage account. You can also use blob storage to archive rarely used data or to store website assets such as images and media.

If you are performing analytics on the data, set up the storage account as an Azure Data Lake Storage Gen2 account by setting the **Hierarchical Namespace** option to **Enabled**. Because Azure Data Lake Storage Gen2 is integrated into the Azure Storage platform, applications can use either the Blob APIs or the Azure Data Lake Storage Gen2 file system APIs to access data.

Next unit: Understand the stages for processing big data

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Understand the stages for processing big data

5 minutes

Data lakes have a fundamental role in a wide range of big data architectures. These architectures can involve the creation of:

- An enterprise data warehouse.
- Advanced analytics against big data.
- A real-time analytical solution.

There are four stages for processing big data solutions that are common to all architectures:

- **Ingest** - The ingestion phase identifies the technology and processes that are used to acquire the source data. This data can come from files, logs, and other types of unstructured data that must be put into the data lake. The technology that is used will vary depending on the frequency that the data is transferred. For example, for batch movement of data, pipelines in Azure Synapse Analytics or Azure Data Factory may be the most appropriate technology to use. For real-time ingestion of data, Apache Kafka for HDInsight or Stream Analytics may be an appropriate choice.
- **Store** - The store phase identifies where the ingested data should be placed. Azure Data Lake Storage Gen2 provides a secure and scalable storage solution that is compatible with commonly used big data processing technologies.
- **Prep and train** - The prep and train phase identifies the technologies that are used to perform data preparation and model training and scoring for machine learning solutions. Common technologies that are used in this phase are Azure Synapse Analytics, Azure Databricks, Azure HDInsight, and Azure Machine Learning.
- **Model and serve** - Finally, the model and serve phase involves the technologies that will present the data to users. These technologies can include visualization tools such as Microsoft Power BI, or analytical data stores such as Azure Synapse Analytics. Often, a combination of multiple technologies will be used depending on the business requirements.

Next unit: Use Azure Data Lake Storage Gen2 in data analytics workloads

100 XP

Use Azure Data Lake Storage Gen2 in data analytics workloads

5 minutes

Azure Data Lake Store Gen2 is an enabling technology for multiple data analytics use cases. Let's explore a few common types of analytical workload, and identify how Azure Data Lake Storage Gen2 works with other Azure services to support them.

Big data processing and analytics

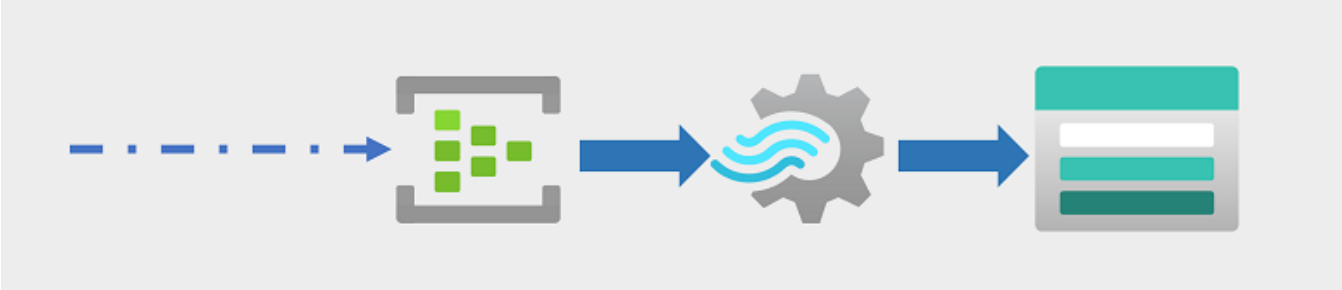
Big data scenarios usually refer to analytical workloads that involve massive *volumes* of data in a *variety* of formats that needs to be processed at a fast *velocity* - the so-called "three v's". Azure Data Lake Storage Gen 2 provides a scalable and secure distributed data store on which big data services such as Azure Synapse Analytics, Azure Databricks, and Azure HDInsight can apply data processing frameworks such as Apache Spark, Hive, and Hadoop. The distributed nature of the storage and the processing compute enables tasks to be performed in parallel, resulting in high-performance and scalability even when processing huge amounts of data.

Data warehousing

Data warehousing has evolved in recent years to integrate large volumes of data stored as files in a data lake with relational tables in a data warehouse. In a typical example of a data warehousing solution, data is extracted from operational data stores, such as Azure SQL database or Azure Cosmos DB, and transformed into structures more suitable for analytical workloads. Often, the data is staged in a data lake in order to facilitate distributed processing before being loaded into a relational data warehouse. In some cases, the data warehouse uses *external* tables to define a relational metadata layer over files in the data lake and create a hybrid "data lakehouse" or "lake database" architecture. The data warehouse can then support analytical queries for reporting and visualization.

There are multiple ways to implement this kind of data warehousing architecture. The diagram shows a solution in which Azure Synapse Analytics hosts *pipelines* to perform *extract, transform, and load* (ETL) processes using Azure Data Factory technology. These processes extract data from operational data sources and load it into a data lake hosted in an Azure Data Lake Storage Gen2 container. The data is then processed and loaded into a relational data warehouse in an Azure Synapse Analytics dedicated SQL pool, from where it can support data visualization and reporting using Microsoft Power BI.

Real-time data analytics



Increasingly, businesses and other organizations need to capture and analyze perpetual streams of data, and analyze it in real-time (or as near to real-time as possible). These streams of data can be generated from connected devices (often referred to as *internet-of-things* or *IoT*

devices) or from data generated by users in social media platforms or other applications. Unlike traditional *batch processing* workloads, streaming data requires a solution that can capture and process a boundless stream of data events as they occur.

Streaming events are often captured in a queue for processing. There are multiple technologies you can use to perform this task, including Azure Event Hubs as shown in the image. From here, the data is processed, often to aggregate data over temporal windows (for example to count the number of social media messages with a given tag every five minutes, or to calculate the average reading of an Internet connected sensor per minute). Azure Stream Analytics enables you to create *jobs* that query and aggregate event data as it arrives, and write the results in an output *sink*. One such sink is Azure Data Lake Storage Gen2; from where the captured real-time data can be analyzed and visualized.

Data science and machine learning

Data science involves the statistical analysis of large volumes of data, often using tools such as Apache Spark and scripting languages such as Python. Azure Data Lake Storage Gen 2 provides a highly scalable cloud-based data store for the volumes of data required in data science workloads.

Machine learning is a subarea of data science that deals with training predictive models. Model training requires huge amounts of data, and the ability to process that data efficiently. Azure Machine Learning is a cloud service in which data scientists can run Python code in notebooks using dynamically allocated distributed compute resources. The compute processes data in Azure Data Lake Storage Gen2 containers to train models, which can then be deployed as production web services to support predictive analytical workloads.

200 XP

Knowledge check

3 minutes

Check your knowledge

1. Azure Data Lake Storage Gen2 stores data in... *

- ○ A document database hosted in Azure Cosmos DB.
- ○ An HDFS-compatible file system hosted in Azure Storage.

✓ **Correct. Azure Data Lake Storage Gen2 stores data in an HDFS compatible file system in an Azure Storage blob container.**

- ○ A relational data warehouse hosted in Azure Synapse Analytics.

2. What option must you enable to use Azure Data Lake Storage Gen2? *

- ○ Global replication
- ○ Data encryption

✗ **Incorrect. data encryption does not enable Azure Data Lake Storage Gen2 containers.**

- ○ Hierarchical namespace

✓ **Correct. To enable Azure Data Lake Storage Gen2 containers, you must turn on the Hierarchical namespace option.**

Next unit: Summary

Continue

100 XP

Summary

1 minute

Azure Data Lake Storage Gen2 provides a cloud storage service that is available, secure, durable, scalable, and redundant. It's a comprehensive data lake solution.

Azure Data Lake Storage brings efficiencies to process big data analytics workloads and can provide data to many compute technologies including Azure Synapse Analytics, Azure HDInsight, and Azure Databricks without needing to move the data around. Creating an Azure Data Lake Storage Gen2 data store can be an important tool in building a big data analytics solution.

Tip

To learn more about Azure Data Lake Storage Gen2, see **Introduction to Azure Data Lake Storage Gen2** in the Microsoft Azure documentation.

Module complete:

Unlock achievement

Introduction to Azure Data Lake Storage Gen2

Article • 03/30/2023

Azure Data Lake Storage Gen2 is a set of capabilities dedicated to big data analytics, built on Azure Blob Storage.

Data Lake Storage Gen2 converges the capabilities of Azure Data Lake Storage Gen1 with Azure Blob Storage. For example, Data Lake Storage Gen2 provides file system semantics, file-level security, and scale. Because these capabilities are built on Blob storage, you also get low-cost, tiered storage, with high availability/disaster recovery capabilities.

Data Lake Storage Gen2 makes Azure Storage the foundation for building enterprise data lakes on Azure. Designed from the start to service multiple petabytes of information while sustaining hundreds of gigabits of throughput, Data Lake Storage Gen2 allows you to easily manage massive amounts of data.

What is a Data Lake?

A *data lake* is a single, centralized repository where you can store all your data, both structured and unstructured. A data lake enables your organization to quickly and more easily store, access, and analyze a wide variety of data in a single location. With a data lake, you don't need to conform your data to fit an existing structure. Instead, you can store your data in its raw or native format, usually as files or as binary large objects (blobs).

Azure Data Lake Storage is a cloud-based, enterprise data lake solution. It's engineered to store massive amounts of data in any format, and to facilitate big data analytical workloads. You use it to capture data of any type and ingestion speed in a single location for easy access and analysis using various frameworks.

Data Lake Storage Gen2

Azure Data Lake Storage Gen2 refers to the current implementation of Azure's Data Lake Storage solution. The previous implementation, *Azure Data Lake Storage Gen1* will be retired on February 29, 2024.

Unlike Data Lake Storage Gen1, Data Lake Storage Gen2 isn't a dedicated service or account type. Instead, it's implemented as a set of capabilities that you use with the Blob

Storage service of your Azure Storage account. You can unlock these capabilities by enabling the hierarchical namespace setting.

Data Lake Storage Gen2 includes the following capabilities.

✓ Hadoop-compatible access

✓ Hierarchical directory structure

✓ Optimized cost and performance

✓ Finer grain security model

✓ Massive scalability

Hadoop-compatible access

Azure Data Lake Storage Gen2 is primarily designed to work with Hadoop and all frameworks that use the Apache Hadoop Distributed File System (HDFS) as their data access layer. Hadoop distributions include the Azure Blob File System (ABFS) driver, which enables many applications and frameworks to access Azure Blob Storage data directly. The ABFS driver is optimized specifically for big data analytics. The corresponding REST APIs are surfaced through the endpoint `dfs.core.windows.net`.

Data analysis frameworks that use HDFS as their data access layer can directly access Azure Data Lake Storage Gen2 data through ABFS. The Apache Spark analytics engine and the Presto SQL query engine are examples of such frameworks.

For more information about supported services and platforms, see Azure services that support Azure Data Lake Storage Gen2 and Open source platforms that support Azure Data Lake Storage Gen2.

Hierarchical directory structure

The hierarchical namespace is a key feature that enables Azure Data Lake Storage Gen2 to provide high-performance data access at object storage scale and price. You can use this feature to organize all the objects and files within your storage account into a hierarchy of directories and nested subdirectories. In other words, your Azure Data Lake Storage Gen2 data is organized in much the same way that files are organized on your computer.

Operations such as renaming or deleting a directory, become single atomic metadata operations on the directory. There's no need to enumerate and process all objects that share the name prefix of the directory.

Optimized cost and performance

Azure Data Lake Storage Gen2 is priced at Azure Blob Storage levels. It builds on Azure Blob Storage capabilities such as automated lifecycle policy management and object level tiering to manage big data storage costs.

Performance is optimized because you don't need to copy or transform data as a prerequisite for analysis. The hierarchical namespace capability of Azure Data Lake Storage allows for efficient access and navigation. This architecture means that data processing requires fewer computational resources, reducing both the speed and cost of accessing data.

Finer grain security model

The Azure Data Lake Storage Gen2 access control model supports both Azure role-based access control (Azure RBAC) and Portable Operating System Interface for UNIX (POSIX) access control lists (ACLs). There are also a few extra security settings that are specific to Azure Data Lake Storage Gen2. You can set permissions either at the directory level or at the file level. All stored data is encrypted at rest by using either Microsoft-managed or customer-managed encryption keys.

Massive scalability

Azure Data Lake Storage Gen2 offers massive storage and accepts numerous data types for analytics. It doesn't impose any limits on account sizes, file sizes, or the amount of data that can be stored in the data lake. Individual files can have sizes that range from a few kilobytes (KBs) to a few petabytes (PBs). Processing is executed at near-constant per-request latencies that are measured at the service, account, and file levels.

This design means that Azure Data Lake Storage Gen2 can easily and quickly scale up to meet the most demanding workloads. It can also just as easily scale back down when demand drops.

Built on Azure Blob Storage

The data that you ingest persist as blobs in the storage account. The service that manages blobs is the Azure Blob Storage service. Data Lake Storage Gen2 describes the capabilities or "enhancements" to this service that caters to the demands of big data analytic workloads.

Because these capabilities are built on Blob Storage, features such as diagnostic logging, access tiers, and lifecycle management policies are available to your account. Most Blob Storage features are fully supported, but some features might be supported only at the preview level and there are a handful of them that aren't yet supported. For a complete list of support statements, see Blob Storage feature support in Azure Storage accounts. The status of each listed feature will change over time as support continues to expand.

Documentation and terminology

The Azure Blob Storage table of contents features two sections of content. The **Data Lake Storage Gen2** section of content provides best practices and guidance for using Data Lake Storage Gen2 capabilities. The **Blob Storage** section of content provides guidance for account features not specific to Data Lake Storage Gen2.

As you move between sections, you might notice some slight terminology differences. For example, content featured in the Blob Storage documentation, will use the term *blob* instead of *file*. Technically, the files that you ingest to your storage account become blobs in your account. Therefore, the term is correct. However, the term *blob* can cause confusion if you're used to the term *file*. You'll also see the term *container* used to refer to a *file system*. Consider these terms as synonymous.

See also

- Introduction to Azure Data Lake Storage Gen2 (Training module)
- Best practices for using Azure Data Lake Storage Gen2
- Known issues with Azure Data Lake Storage Gen2
- Multi-protocol access on Azure Data Lake Storage

Best practices for using Azure Data Lake Storage Gen2

Article • 03/09/2023

This article provides best practice guidelines that help you optimize performance, reduce costs, and secure your Data Lake Storage Gen2 enabled Azure Storage account.

For general suggestions around structuring a data lake, see these articles:

- Overview of Azure Data Lake Storage for the data management and analytics scenario
- Provision three Azure Data Lake Storage Gen2 accounts for each data landing zone

Find documentation

Azure Data Lake Storage Gen2 isn't a dedicated service or account type. It's a set of capabilities that support high throughput analytic workloads. The Data Lake Storage Gen2 documentation provides best practices and guidance for using these capabilities. For all other aspects of account management such as setting up network security, designing for high availability, and disaster recovery, see the Blob storage documentation content.

Evaluate feature support and known issues

Use the following pattern as you configure your account to use Blob storage features.

1. Review the Blob Storage feature support in Azure Storage accounts article to determine whether a feature is fully supported in your account. Some features aren't yet supported or have partial support in Data Lake Storage Gen2 enabled accounts. Feature support is always expanding so make sure to periodically review this article for updates.

2. Review the Known issues with Azure Data Lake Storage Gen2 article to see if there are any limitations or special guidance around the feature you intend to use.

3. Scan feature articles for any guidance that is specific to Data Lake Storage Gen2 enabled accounts.

Understand the terms used in documentation

As you move between content sets, you notice some slight terminology differences. For example, content featured in the Blob storage documentation, will use the term *blob* instead of *file*. Technically, the files that you ingest to your storage account become blobs in your account. Therefore, the term is correct. However, the term *blob* can cause confusion if you're used to the term *file*. You'll also see the term *container* used to refer to a *file system*. Consider these terms as synonymous.

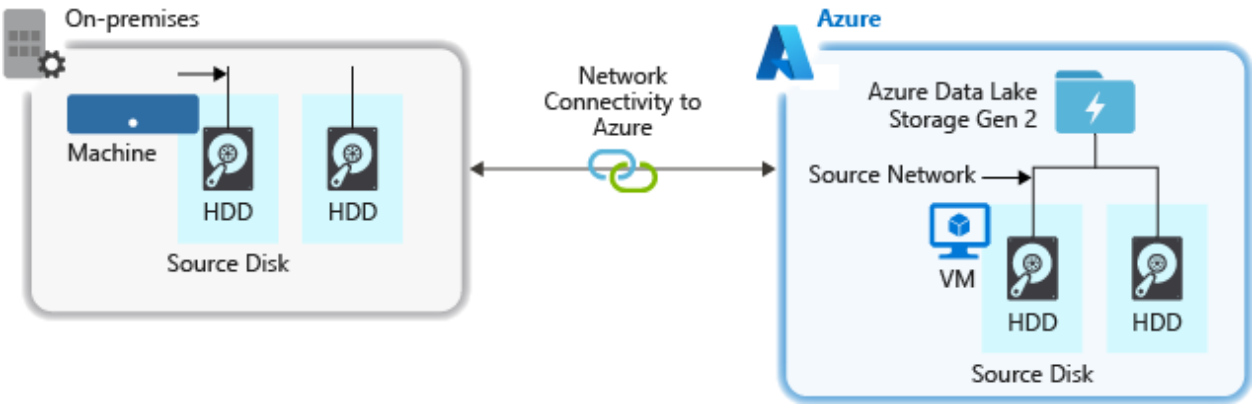
Consider premium

If your workloads require a low consistent latency and/or require a high number of input output operations per second (IOP), consider using a premium block blob storage account. This type of account makes data available via high-performance hardware. Data is stored on solid-state drives (SSDs) which are optimized for low latency. SSDs provide higher throughput compared to traditional hard drives. The storage costs of premium performance are higher, but transaction costs are lower. Therefore, if your workloads execute a large number of transactions, a premium performance block blob account can be economical.

If your storage account is going to be used for analytics, we highly recommend that you use Azure Data Lake Storage Gen2 along with a premium block blob storage account. This combination of using premium block blob storage accounts along with a Data Lake Storage enabled account is referred to as the premium tier for Azure Data Lake Storage.

Optimize for data ingest

When ingesting data from a source system, the source hardware, source network hardware, or the network connectivity to your storage account can be a bottleneck.



Source hardware

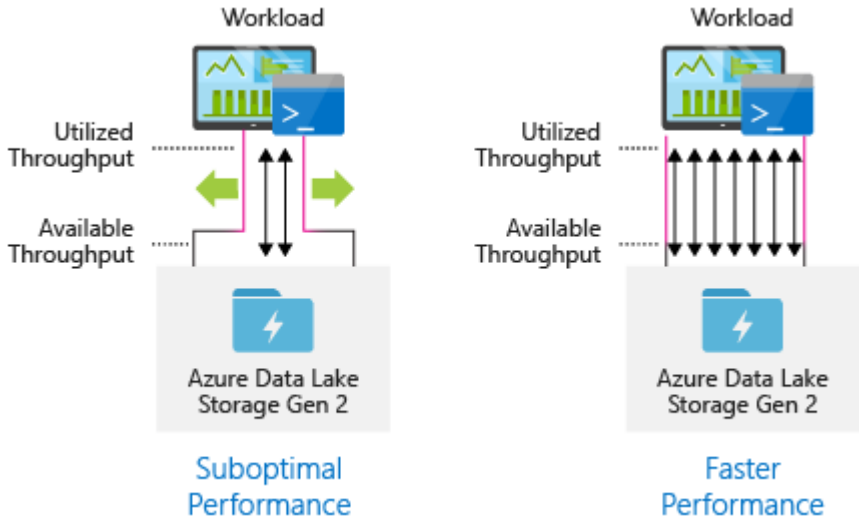
Whether you're using on-premises machines or Virtual Machines (VMs) in Azure, make sure to carefully select the appropriate hardware. For disk hardware, consider using Solid State Drives (SSD) and pick disk hardware that has faster spindles. For network hardware, use the fastest Network Interface Controllers (NIC) as possible. On Azure, we recommend Azure D14 VMs, which have the appropriately powerful disk and networking hardware.

Network connectivity to the storage account

The network connectivity between your source data and your storage account can sometimes be a bottleneck. When your source data is on premise, consider using a dedicated link with Azure ExpressRoute . If your source data is in Azure, the performance is best when the data is in the same Azure region as your Data Lake Storage Gen2 enabled account.

Configure data ingestion tools for maximum parallelization

To achieve the best performance, use all available throughput by performing as many reads and writes in parallel as possible.



The following table summarizes the key settings for several popular ingestion tools.

Tool	Settings
DistCp	-m (mapper)
Azure Data Factory	parallelCopies
Sqoop	fs.azure.block.size, -m (mapper)

> **Note**
>
> The overall performance of your ingest operations depend on other factors that are specific to the tool that you're using to ingest data. For the best up-to-date guidance, see the documentation for each tool that you intend to use.

Your account can scale to provide the necessary throughput for all analytics scenarios. By default, a Data Lake Storage Gen2 enabled account provides enough throughput in its default configuration to meet the needs of a broad category of use cases. If you run into the default limit, the account can be configured to provide more throughput by contacting Azure Support .

Structure data sets

Consider pre-planning the structure of your data. File format, file size, and directory structure can all impact performance and cost.

File formats

Data can be ingested in various formats. Data can appear in human readable formats such as JSON, CSV, or XML or as compressed binary formats such as `.tar.gz`. Data can come in various sizes as well. Data can be composed of large files (a few terabytes) such as data from an export of a SQL table from your on-premises systems. Data can also come in the form of a large number of tiny files (a few kilobytes) such as data from real-time events from an Internet of things (IoT) solution. You can optimize efficiency and costs by choosing an appropriate file format and file size.

Hadoop supports a set of file formats that are optimized for storing and processing structured data. Some common formats are Avro, Parquet, and Optimized Row Columnar (ORC) format. All of these formats are machine-readable binary file formats. They're compressed to help you manage file size. They have a schema embedded in each file, which makes them self-describing. The difference between these formats is in how data is stored. Avro stores data in a row-based format and the Parquet and ORC formats store data in a columnar format.

Consider using the Avro file format in cases where your I/O patterns are more write heavy, or the query patterns favor retrieving multiple rows of records in their entirety. For example, the Avro format works well with a message bus such as Event Hubs or Kafka that write multiple events/messages in succession.

Consider Parquet and ORC file formats when the I/O patterns are more read heavy or when the query patterns are focused on a subset of columns in the records. Read transactions can be optimized to retrieve specific columns instead of reading the entire record.

Apache Parquet is an open source file format that is optimized for read heavy analytics pipelines. The columnar storage structure of Parquet lets you skip over non-relevant data. Your queries are much more efficient because they can narrowly scope which data to send from storage to the analytics engine. Also, because similar data types (for a column) are stored together, Parquet supports efficient data compression and encoding schemes that can lower data storage costs. Services such as Azure Synapse Analytics, Azure Databricks and Azure Data Factory have native functionality that take advantage of Parquet file formats.

File size

Larger files lead to better performance and reduced costs.

Typically, analytics engines such as HDInsight have a per-file overhead that involves tasks such as listing, checking access, and performing various metadata operations. If you store your data as many small files, this can negatively affect performance. In general, organize your data into larger sized files for better performance (256 MB to 100 GB in size). Some engines and applications might have trouble efficiently processing files that are greater than 100 GB in size.

Increasing file size can also reduce transaction costs. Read and write operations are billed in 4 megabyte increments so you're charged for operation whether or not the file contains 4 megabytes or only a few kilobytes. For pricing information, see Azure Data Lake Storage pricing .

Sometimes, data pipelines have limited control over the raw data, which has lots of small files. In general, we recommend that your system have some sort of process to aggregate small files into larger ones for use by downstream applications. If you're processing data in real time, you can use a real time streaming engine (such as Azure Stream Analytics or Spark Streaming) together with a message broker (such as Event Hubs or Apache Kafka) to store your data as larger files. As you aggregate small files into larger ones, consider saving them in a read-optimized format such as Apache Parquet for downstream processing.

Directory structure

Every workload has different requirements on how the data is consumed, but these are some common layouts to consider when working with Internet of Things (IoT), batch scenarios or when optimizing for time-series data.

IoT structure

In IoT workloads, there can be a great deal of data being ingested that spans across numerous products, devices, organizations, and customers. It's important to pre-plan the directory layout for organization, security, and efficient processing of the data for down-stream consumers. A general template to consider might be the following layout:

- *{Region}/{SubjectMatter(s)}/{yyyy}/{mm}/{dd}/{hh}/*

For example, landing telemetry for an airplane engine within the UK might look like the following structure:

- *UK/Planes/BA1293/Engine1/2017/08/11/12/*

In this example, by putting the date at the end of the directory structure, you can use ACLs to more easily secure regions and subject matters to specific users and groups. If you put the date structure at the beginning, it would be much more difficult to secure these regions and subject matters. For example, if you wanted to provide access only to UK data or certain planes, you'd need to apply a separate permission for numerous directories under every hour directory. This structure would also exponentially increase the number of directories as time went on.

Batch jobs structure

A commonly used approach in batch processing is to place data into an "in" directory. Then, once the data is processed, put the new data into an "out" directory for downstream processes to consume. This directory structure is sometimes used for jobs that require processing on individual files, and might not require massively parallel processing over large datasets. Like the IoT structure recommended above, a good directory structure has the parent-level directories for things such as region and subject matters (for example, organization, product, or producer). Consider date and time in the structure to allow better organization, filtered searches, security, and automation in the processing. The level of granularity for the date structure is determined by the interval on which the data is uploaded or processed, such as hourly, daily, or even monthly.

Sometimes file processing is unsuccessful due to data corruption or unexpected formats. In such cases, a directory structure might benefit from a **/bad** folder to move the files to for further inspection. The batch job might also handle the reporting or

notification of these *bad* files for manual intervention. Consider the following template structure:

- *{Region}/{SubjectMatter(s)}/In/{yyyy}/{mm}/{dd}/{hh}/*
- *{Region}/{SubjectMatter(s)}/Out/{yyyy}/{mm}/{dd}/{hh}/*
- *{Region}/{SubjectMatter(s)}/Bad/{yyyy}/{mm}/{dd}/{hh}/*

For example, a marketing firm receives daily data extracts of customer updates from their clients in North America. It might look like the following snippet before and after being processed:

- *NA/Extracts/ACMEPaperCo/In/2017/08/14/updates_08142017.csv*
- *NA/Extracts/ACMEPaperCo/Out/2017/08/14/processed_updates_08142017.csv*

In the common case of batch data being processed directly into databases such as Hive or traditional SQL databases, there isn't a need for an **/in** or **/out** directory because the output already goes into a separate folder for the Hive table or external database. For example, daily extracts from customers would land into their respective directories. Then, a service such as Azure Data Factory, Apache Oozie , or Apache Airflow would trigger a daily Hive or Spark job to process and write the data into a Hive table.

Time series data structure

For Hive workloads, partition pruning of time-series data can help some queries read only a subset of the data, which improves performance.

Those pipelines that ingest time-series data, often place their files with a structured naming for files and folders. Below is a common example we see for data that is structured by date:

/DataSet/YYYY/MM/DD/datafile_YYYY_MM_DD.tsv

Notice that the datetime information appears both as folders and in the filename.

For date and time, the following is a common pattern

/DataSet/YYYY/MM/DD/HH/mm/datafile_YYYY_MM_DD_HH_mm.tsv

Again, the choice you make with the folder and file organization should optimize for the larger file sizes and a reasonable number of files in each folder.

Set up security

Start by reviewing the recommendations in the Security recommendations for Blob storage article. You'll find best practice guidance about how to protect your data from accidental or malicious deletion, secure data behind a firewall, and use Azure Active Directory (Azure AD) as the basis of identity management.

Then, review the Access control model in Azure Data Lake Storage Gen2 article for guidance that is specific to Data Lake Storage Gen2 enabled accounts. This article helps you understand how to use Azure role-based access control (Azure RBAC) roles together with access control lists (ACLs) to enforce security permissions on directories and files in your hierarchical file system.

Ingest, process, and analyze

There are many different sources of data and different ways in which that data can be ingested into a Data Lake Storage Gen2 enabled account.

For example, you can ingest large sets of data from HDInsight and Hadoop clusters or smaller sets of *ad hoc* data for prototyping applications. You can ingest streamed data that is generated by various sources such as applications, devices, and sensors. For this type of data, you can use tools to capture and process the data on an event-by-event basis in real time, and then write the events in batches into your account. You can also ingest web server logs, which contain information such as the history of page requests. For log data, consider writing custom scripts or applications to upload them so that you'll have the flexibility to include your data uploading component as part of your larger big data application.

Once the data is available in your account, you can run analysis on that data, create visualizations, and even download data to your local machine or to other repositories such as an Azure SQL database or SQL Server instance.

The following table recommends tools that you can use to ingest, analyze, visualize, and download data. Use the links in this table to find guidance about how to configure and use each tool.

Purpose	**Tools & Tool guidance**
Ingest ad hoc data	Azure portal, Azure PowerShell, Azure CLI, REST, Azure Storage Explorer , Apache DistCp, AzCopy
Ingest relational data	Azure Data Factory
Ingest web server logs	Azure PowerShell, Azure CLI, REST, Azure SDKs (.NET, Java, Python, and Node.js), Azure Data Factory

Purpose	Tools & Tool guidance
Ingest from HDInsight clusters	Azure Data Factory, Apache DistCp, AzCopy
Ingest from Hadoop clusters	Azure Data Factory, Apache DistCp, WANdisco LiveData Migrator for Azure, Azure Data Box
Ingest large data sets (several terabytes)	Azure ExpressRoute
Process & analyze data	Azure Synapse Analytics, Azure HDInsight, Databricks
Visualize data	Power BI, Azure Data Lake Storage query acceleration
Download data	Azure portal, PowerShell, Azure CLI, REST, Azure SDKs (.NET, Java, Python, and Node.js), Azure Storage Explorer, AzCopy, Azure Data Factory, Apache DistCp

> **Note**
>
> This table doesn't reflect the complete list of Azure services that support Data Lake Storage Gen2. To see a list of supported Azure services, their level of support, see **Azure services that support Azure Data Lake Storage Gen2**.

Monitor telemetry

Monitoring the use and performance is an important part of operationalizing your service. Examples include frequent operations, operations with high latency, or operations that cause service-side throttling.

All of the telemetry for your storage account is available through Azure Storage logs in Azure Monitor. This feature integrates your storage account with Log Analytics and Event Hubs, while also enabling you to archive logs to another storage account. To see the full list of metrics and resources logs and their associated schema, see Azure Storage monitoring data reference.

Where you choose to store your logs depends on how you plan to access them. For example, if you want to access your logs in near real time, and be able to correlate events in logs with other metrics from Azure Monitor, you can store your logs in a Log Analytics workspace. Then, query your logs by using KQL and author queries, which enumerate the `StorageBlobLogs` table in your workspace.

If you want to store your logs for both near real-time query and long term retention, you can configure your diagnostic settings to send logs to both a Log Analytics workspace and a storage account.

If you want to access your logs through another query engine such as Splunk, you can configure your diagnostic settings to send logs to an event hub and ingest logs from the event hub to your chosen destination.

Azure Storage logs in Azure Monitor can be enabled through the Azure portal, PowerShell, the Azure CLI, and Azure Resource Manager templates. For at-scale deployments, Azure Policy can be used with full support for remediation tasks. For more information, see ciphertxt/AzureStoragePolicy .

See also

- Key considerations for Azure Data Lake Storage
- Access control model in Azure Data Lake Storage Gen2
- The hitchhiker's guide to the Data Lake
- Overview of Azure Data Lake Storage Gen2

Known issues with Azure Data Lake Storage Gen2

Article • 03/10/2023

This article describes limitations and known issues for accounts that have the hierarchical namespace feature enabled.

> **Note**
>
> Some of the features described in this article might not be supported in accounts that have Network File System (NFS) 3.0 support enabled. To view a table that shows the impact of feature support when various capabilities are enabled, see **Blob Storage feature support in Azure Storage accounts**.

Supported Blob storage features

An increasing number of Blob storage features now work with accounts that have a hierarchical namespace. For a complete list, see Blob Storage features available in Azure Data Lake Storage Gen2.

Supported Azure service integrations

Azure Data Lake Storage Gen2 supports several Azure services that you can use to ingest data, perform analytics, and create visual representations. For a list of supported Azure services, see Azure services that support Azure Data Lake Storage Gen2.

For more information, see Azure services that support Azure Data Lake Storage Gen2.

Supported open source platforms

Several open source platforms support Data Lake Storage Gen2. For a complete list, see Open source platforms that support Azure Data Lake Storage Gen2.

For more information, see Open source platforms that support Azure Data Lake Storage Gen2.

Blob storage APIs

Data Lake Storage Gen2 APIs, NFS 3.0, and Blob APIs can operate on the same data.

This section describes issues and limitations with using blob APIs, NFS 3.0, and Data Lake Storage Gen2 APIs to operate on the same data.

- You can't use blob APIs, NFS 3.0, and Data Lake Storage APIs to write to the same instance of a file. If you write to a file by using Data Lake Storage Gen2 APIs or NFS 3.0, then that file's blocks won't be visible to calls to the Get Block List blob API. The only exception is when you're overwriting. You can overwrite a file/blob using either API or with NFS 3.0 by using the zero-truncate option.

- When you use the List Blobs operation without specifying a delimiter, the results include both directories and blobs. If you choose to use a delimiter, use only a forward slash (`/`). This is the only supported delimiter.

- If you use the Delete Blob API to delete a directory, that directory is deleted only if it's empty. This means that you can't use the Blob API delete directories recursively.

These Blob REST APIs aren't supported:

- Put Blob (Page)
- Put Page
- Get Page Ranges
- Incremental Copy Blob
- Put Page from URL

Unmanaged VM disks aren't supported in accounts that have a hierarchical namespace. If you want to enable a hierarchical namespace on a storage account, place unmanaged VM disks into a storage account that doesn't have the hierarchical namespace feature enabled.

Support for setting access control lists (ACLs) recursively

The ability to apply ACL changes recursively from parent directory to child items is generally available. In the current release of this capability, you can apply ACL changes by using Azure Storage Explorer, PowerShell, Azure CLI, and the .NET, Java, and Python SDK. Support isn't yet available for the Azure portal.

Access control lists (ACL) and anonymous read access

If anonymous read access has been granted to a container, then ACLs have no effect on that container or the files in that container. This only affects read requests. Write requests will still honor the ACLs. We recommend requiring authorization for all requests to blob data.

AzCopy

Use only the latest version of AzCopy (AzCopy v10). Earlier versions of AzCopy such as AzCopy v8.1, aren't supported.

Azure Storage Explorer

Use only versions `1.6.0` or higher.

Storage browser in the Azure portal

In the storage browser that appears in the Azure portal, you can't access a file or folder by specifying a path. Instead, you must browse through folders to reach a file. Therefore, if an ACL grants a user read access to a file but not read access to all folders leading up to the file, that user won't be able to view the file in storage browser.

Third party applications

Third party applications that use REST APIs to work will continue to work if you use them with Data Lake Storage Gen2. Applications that call Blob APIs will likely work.

Windows Azure Storage Blob (WASB) driver

Currently, the WASB driver, which was designed to work with the Blob API only, encounters problems in a few common scenarios. Specifically, when it's a client to a hierarchical namespace enabled storage account. Multi-protocol access on Data Lake Storage won't mitigate these issues.

Using the WASB driver as a client to a hierarchical namespace enabled storage account isn't supported. Instead, we recommend that you use the Azure Blob File System (ABFS) driver in your Hadoop environment. If you're trying to migrate off of an on-premises Hadoop environment with a version earlier than Hadoop branch-3, then please open an Azure Support ticket so that we can get in touch with you on the right path forward for you and your organization.

Soft delete for blobs capability

If parent directories for soft-deleted files or directories are renamed, the soft-deleted items may not be displayed correctly in the Azure portal. In such cases you can use PowerShell or Azure CLI to list and restore the soft-deleted items.

Events

If your account has an event subscription, read operations on the secondary endpoint will result in an error. To resolve this issue, remove event subscriptions. Using the Data Lake Storage endpoint (abfss://URI) for non-hierarchical namespace enabled accounts won't generate events, but the blob endpoint (wasb:// URI) will generate events.

> **Tip**
>
> Read access to the secondary endpoint is available only when you enable read-access geo-redundant storage (RA-GRS) or read-access geo-zone-redundant storage (RA-GZRS).

Multi-protocol access on Azure Data Lake Storage

Article • 03/09/2023

Blob APIs work with accounts that have a hierarchical namespace. This unlocks the ecosystem of tools, applications, and services, as well as several Blob storage features to accounts that have a hierarchical namespace.

Until recently, you might have had to maintain separate storage solutions for object storage and analytics storage. That's because Azure Data Lake Storage Gen2 had limited ecosystem support. It also had limited access to Blob service features such as diagnostic logging. A fragmented storage solution is hard to maintain because you have to move data between accounts to accomplish various scenarios. You no longer have to do that.

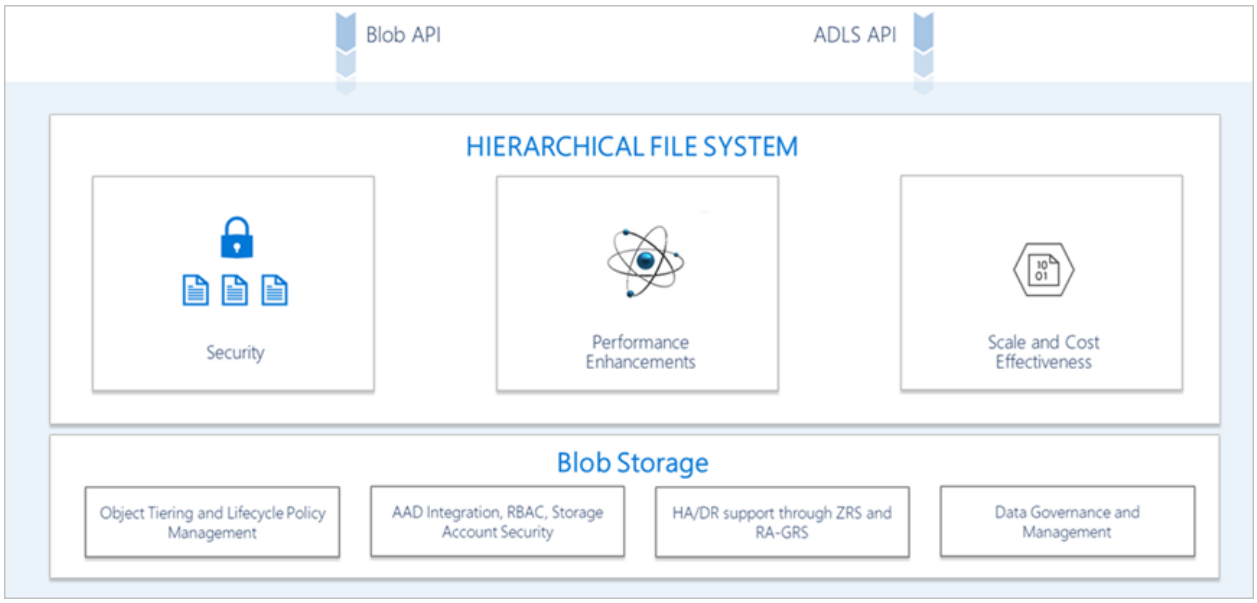
With multi-protocol access on Data Lake Storage, you can work with your data by using the ecosystem of tools, applications, and services. This also includes third-party tools and applications. You can point them to accounts that have a hierarchical namespace without having to modify them. These applications work *as is* even if they call Blob APIs, because Blob APIs can now operate on data in accounts that have a hierarchical namespace.

Blob storage features such as diagnostic logging, access tiers, and Blob storage lifecycle management policies now work with accounts that have a hierarchical namespace. Therefore, you can enable hierarchical namespaces on your blob Storage accounts without losing access to these important features.

> **Note**
>
> Some Azure services or Blob storage features enabled by multi-protocol access remain in preview. These articles summarize the current support for Blob storage features and Azure service integrations.
>
> **Blob Storage feature support in Azure Storage accounts**
>
> **Azure services that support Azure Data Lake Storage Gen2**

How multi-protocol access on data lake storage works

Blob APIs and Data Lake Storage Gen2 APIs can operate on the same data in storage accounts that have a hierarchical namespace. Data Lake Storage Gen2 routes Blob APIs through the hierarchical namespace so that you can get the benefits of first class directory operations and POSIX-compliant access control lists (ACLs).



Existing tools and applications that use the Blob API gain these benefits automatically. Developers won't have to modify them. Data Lake Storage Gen2 consistently applies directory and file-level ACLs regardless of the protocol that tools and applications use to access the data.

See also

- Blob Storage feature support in Azure Storage accounts
- Azure services that support Azure Data Lake Storage Gen2
- Open source platforms that support Azure Data Lake Storage Gen2
- Known issues with Azure Data Lake Storage Gen2

Azure Data Lake Storage key considerations

Article • 09/04/2023

Learn about key storage considerations for your Azure data lakes.

Lifecycle management

Azure Storage offers different access tiers, which allows you to store blob object data in the most cost-effective manner possible. The available access tiers include:

- **Hot:** Optimized for storing data that's accessed frequently.
- **Cool:** Optimized for storing data that's infrequently accessed. Data is stored for at least 30 days.
- **Cold tier:** Optimized for storing data that is infrequently accessed or modified. Data is stored for at least 90 days. The cold tier has lower storage costs and higher access costs compared to the cool tier.
- **Archive:** Optimized for storing data that's rarely accessed. The data is stored for at least 180 days with flexible latency requirements, on the order of hours.

Consider the following information when using access tiers:

- Only the Hot and Cool access tiers can be set at the account level. The Archive access tier isn't available at the account level.
- Hot, Cool, and Archive tiers can all be set at the blob level during upload or after upload.
- Data in the Cool tier has slightly lower availability, but offers the same high durability, retrieval latency, and throughput characteristics as Hot tier data. For data in the Cool tier, slightly lower availability and higher access costs can be acceptable trade-offs for lower overall storage costs compared to the Hot tier.
- Archive storage stores data offline and offers the lowest storage costs. However, it also carries the highest data rehydration and access costs.

For more information, see Hot, Cool and Archive access tiers for blob data.

⊗ **Caution**

For cloud-scale analytics, we recommend that you implement **lifecycle management** using a custom microservice and carefully consider the impact of moving user discoverable data to cool storage.

You should only move sections of your data lake to cool tier for well understood workloads.

Data lakes connectivity

Each of your data lakes should use private endpoints injected into the virtual network of your data landing zone. To provide access across landing zones, connect your data landing zones through virtual network peering. This connection provides an optimal solution from both a cost and access control perspective.

For more information, see Private endpoints and Data management landing zone to data landing zone.

> **Important**
>
> Data from a data landing zone can be accessed from another data landing zone over virtual network peering between the zones. This is done using the private endpoints associated with each data lake account. We recommend turning off all public access to your lakes and using private endpoints. Your platform operations team should control network connectivity across your data landing zones.

Soft delete for containers

Soft delete for containers protects your data from accidental or malicious deletion. If you enable container soft delete for your storage account, deleted containers and their contents are retained in Azure Storage for a length of time you choose. During the data retention period, you can restore previously deleted containers. Restoring a container also restores any blobs that were within that container when it was deleted.

Enable the following data protection features to achieve end-to-end blob data protection:

- Container soft delete, to restore a container that has been deleted. To learn how to enable container soft delete, see Enable and manage soft delete for containers.
- Blob soft delete, to restore a blob or version that has been deleted. To learn how to enable blob soft delete, see Enable and manage soft delete for blobs.

> ⚠ **Warning**
>
> Deleting a storage account can't be undone. Container soft delete does not protect against storage account deletion, only against the deletion of containers within an account. To protect a storage account from deletion, configure a lock on the storage account resource. For more information about locking Azure Resource Manager resources, see **Lock resources to prevent unexpected changes**.

Monitoring

In a data landing zone, all monitoring should be sent to your enterprise-scale management subscription for analysis.

To learn about the monitoring data Azure Storage uses, see Monitoring Azure resources with Azure Monitor. For more information on the logs and metrics Azure Storage creates, see Monitoring Azure Blob Storage.

Log entries are only created if requests are made against the service endpoint. The types of authenticated requests logged are:

- Successful requests
- Failed requests, including timeout, throttling, network, authorization, and other errors
- Requests that use a shared access signature (SAS) or OAuth, including failed and successful requests
- Requests to analytics data, like classic log data in the `$logs` container and class metric data in the `$metric` tables

Requests made by the storage service itself, like log creation or deletion, aren't logged. The types of anonymous requests logged are:

- Successful requests
- Server errors
- Time out errors for both client and server
- Failed HTTP GET requests with the error code 304 (`Not Modified`)

All other failed anonymous requests aren't logged.

> ⓘ **Important**

Set your default monitoring policy to audit storage and send logs to your enterprise-scale management subscription.

Next steps

- Access control and data lake configurations in Azure Data Lake Storage Gen2

Access control and data lake configurations in Azure Data Lake Storage Gen2

Article • 08/29/2023

This article helps you assess and understand the access control mechanisms in Azure Data Lake Storage Gen2. These mechanisms include Azure role-based access control (RBAC) and access control lists (ACLs). You learn:

- How to evaluate access between Azure RBAC and ACLs
- How to configure access control using one or both of these mechanisms
- How to apply the access control mechanisms to data lake implementation patterns

You need a basic knowledge of storage containers, security groups, Azure RBAC, and ACLs. To frame the discussion, we reference a generic data lake structure of raw, enriched, and curated zones.

You can use this document with Data access management.

Use the built-in Azure RBAC roles

Azure Storage has two layers of access: service management and data. You can access subscriptions and storage accounts through the service management layer. Access containers, blobs, and other data resources through the data layer. For example, if you want a list of storage accounts from Azure, send a request to the management endpoint. If you want a list of file systems, folders, or files in a storage account, send a request to a service endpoint.

Roles can contain permissions for management or data layer access. The Reader role grants read-only access to management layer resources but not read access to data.

Roles such as Owner, Contributor, Reader, and Storage Account Contributor, permit a security principal to manage a storage account. But they don't provide access to the data in that account. Only roles explicitly defined for data access permit a security principal to access the data. These roles, except for Reader, get access to the storage keys to access data.

Built-in management roles

Following are the built-in management roles.

- Owner: Manage everything, including access to resources. This role provides key access.
- Contributor: Manage everything, except access to resources. This role provides key access.
- Storage Account Contributor: Full management of storage accounts. This role provides key access.
- Reader: Read and list resources. This role doesn't provide key access.

Built-in data roles

Following are the built-in data roles.

- Storage Blob Data Owner: Full access to Azure Storage blob containers and data, including setting ownership and managing POSIX access control.
- Storage Blob Data Contributor: Read, write, and delete Azure Storage containers and blobs.
- Storage Blob Data Reader: Read and list Azure Storage containers and blobs.

The Storage Blob Data Owner is a super-user role that's granted full access to all mutating operations. These operations include setting the owner of a directory or file and ACLs for directories and files for which they aren't the owner. Super-user access is the only authorized manner to change the owner of a resource.

> **Note**
>
> Azure RBAC assignments can take up to five minutes to propagate and take effect.

How access is evaluated

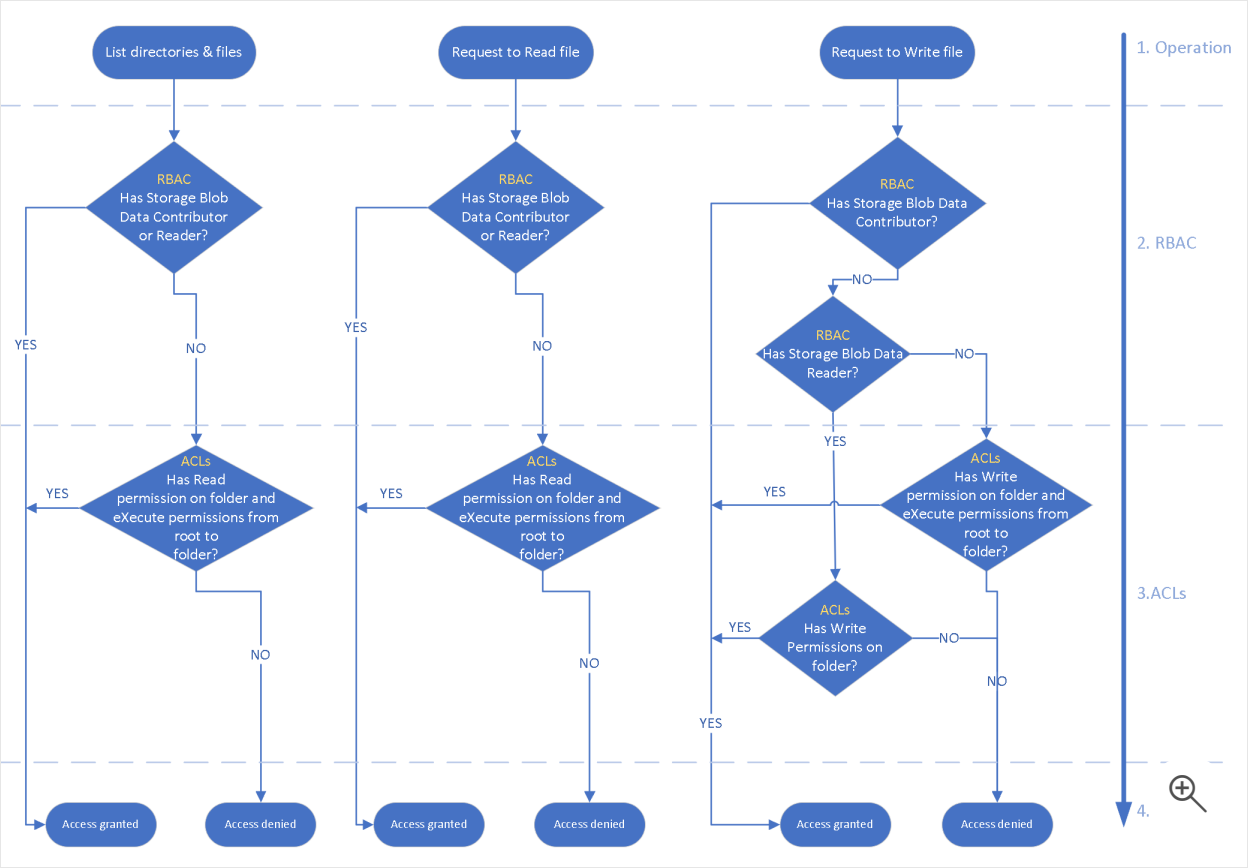
During security principal-based authorization, the system evaluates permissions in the following order. For more information, see the following diagram.

- Azure RBAC is evaluated first and takes priority over any ACL assignments.
- If the operation is fully authorized based on RBAC, ACLs aren't evaluated at all.
- If the operation isn't fully authorized, ACLs are evaluated.

For more information, see How permissions are evaluated.

> **Note**

This permission model applies to Azure Data Lake Storage only. It doesn't apply to general purpose or blob storage without hierarchical namespace enabled. This description excludes shared key and SAS authentication methods. It also excludes scenarios in which the security principal is assigned the Storage Blob Data Owner built-in role, which provides super-user access. Set `allowSharedKeyAccess` to false so that access is audited by identity.



For more information about what ACL-based permissions are required for a given operation, see Access control lists in Azure Data Lake Storage Gen2.

Note

- Access control lists apply only to security principals in the same tenant, including guest users.
- Any user with permissions to attach to a cluster can create Azure Databricks mount points. Configure the mount point using service principal credentials or the Azure AD passthrough option. At the time of creation, permissions are not evaluated. Permissions are evaluated when an operation uses the mount point. Any user who can attach to a cluster can attempt to use the mount point.

- When a user creates a table definition in Azure Databricks or Azure Synapse Analytics, they need to have read access to the underlying data.

Configure access to Azure Data Lake Storage

Set up access control in Azure Data Lake Storage using Azure RBAC, ACLs, or a combination of both.

Configure access using Azure RBAC only

If container-level access control suffices, Azure RBAC assignments offer a simple management approach to securing data. It's recommended that you use access control lists for a large number of restricted data assets or where you require granular access control.

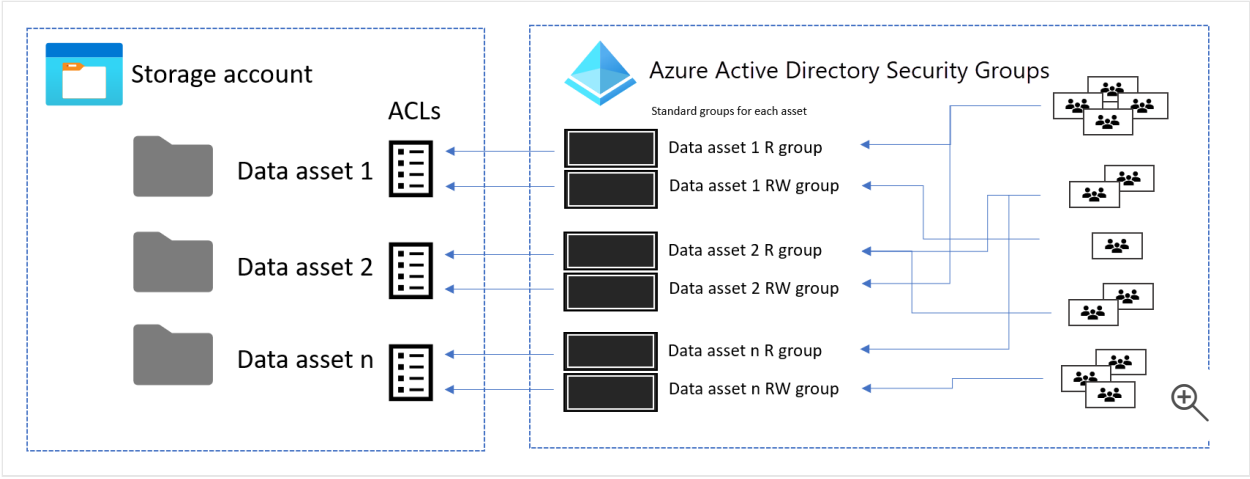
Configure access using ACLs only

The following are access control lists configuration recommendations for cloud-scale analytics.

Assign access control entries to a security group rather than an individual user or service principal. For more information, see Use security groups versus individual users.

When you add or remove users from the group, you aren't required to make updates to Data Lake Storage. Also, using groups reduces the chance of exceeding the 32 access control entries per file or folder ACL. After the four default entries, there are only 28 remaining entries for permission assignments.

Even when you use groups, you could have many access control entries at top levels of the directory tree. This situation happens when granular permissions for different groups are required.



Configure access using both Azure RBAC and access control lists

The Storage Blob Data Contributor and Storage Blob Data Reader permissions provide access to the data and not the storage account. You can grant access at the storage account level or container level. If Storage Blob Data Contributor is assigned, ACLs can't be used to manage access. Where Storage Blob Data Reader is assigned, you can grant elevated write permissions using ACLs. For more information, see How access is evaluated.

This approach favors scenarios where most users need read access but only a few users need write access. The data lake zones could be different storage accounts and data assets could be different containers. The data lake zones could be represented by containers and data assets represented by folders.

Nested access control list group approaches

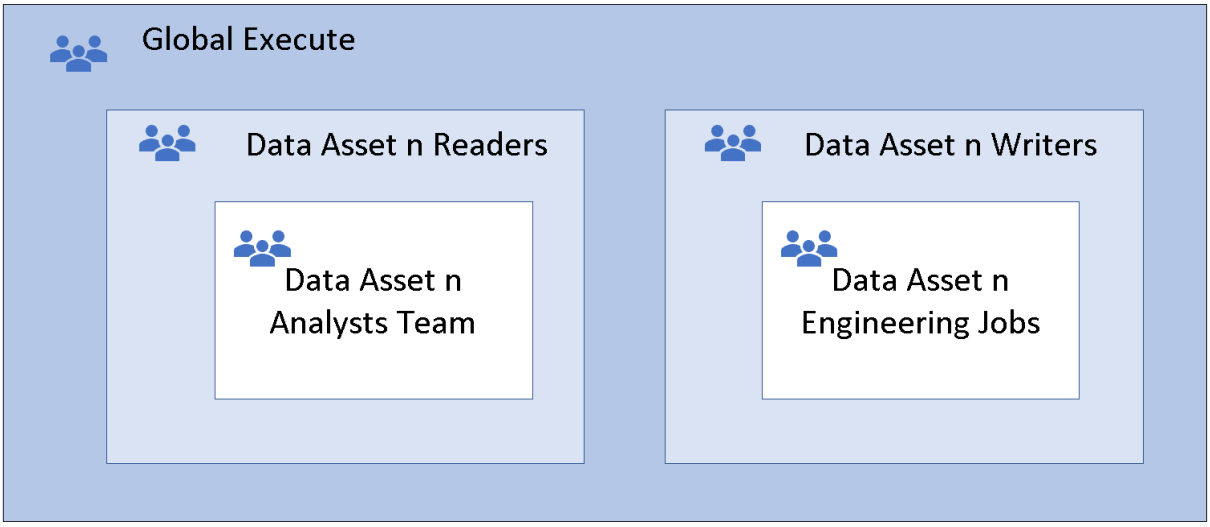
There are two approaches for nested ACL groups.

Option 1: The parent execute group

Before you create files and folders, begin with a parent group. Assign that group run permissions to both default and access ACLs at the container level. Then add the groups that require data access to the parent group.

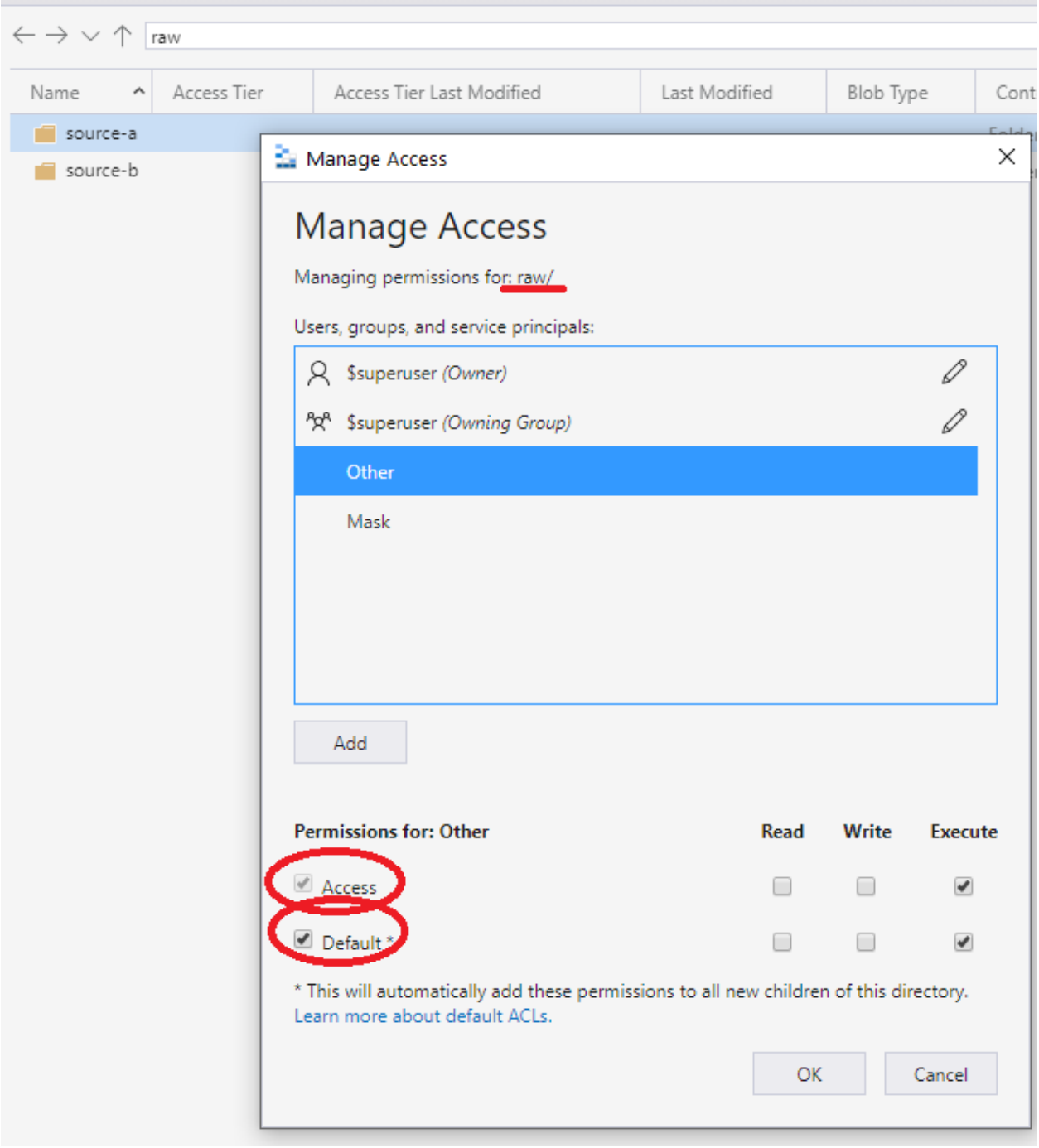
> ⚠ **Warning**
>
> We would recommend against this pattern where you have recursive deletes and instead use **Option 2: The access control list other entry**.

This technique is known as nesting groups. The member group inherits the permissions of the parent group, which provides global run permissions to all member groups. The member group doesn't need run permissions because these permissions are inherited. More nesting might provide greater flexibility and agility. Add security groups that represent teams or automated jobs to the data access reader and writer groups.

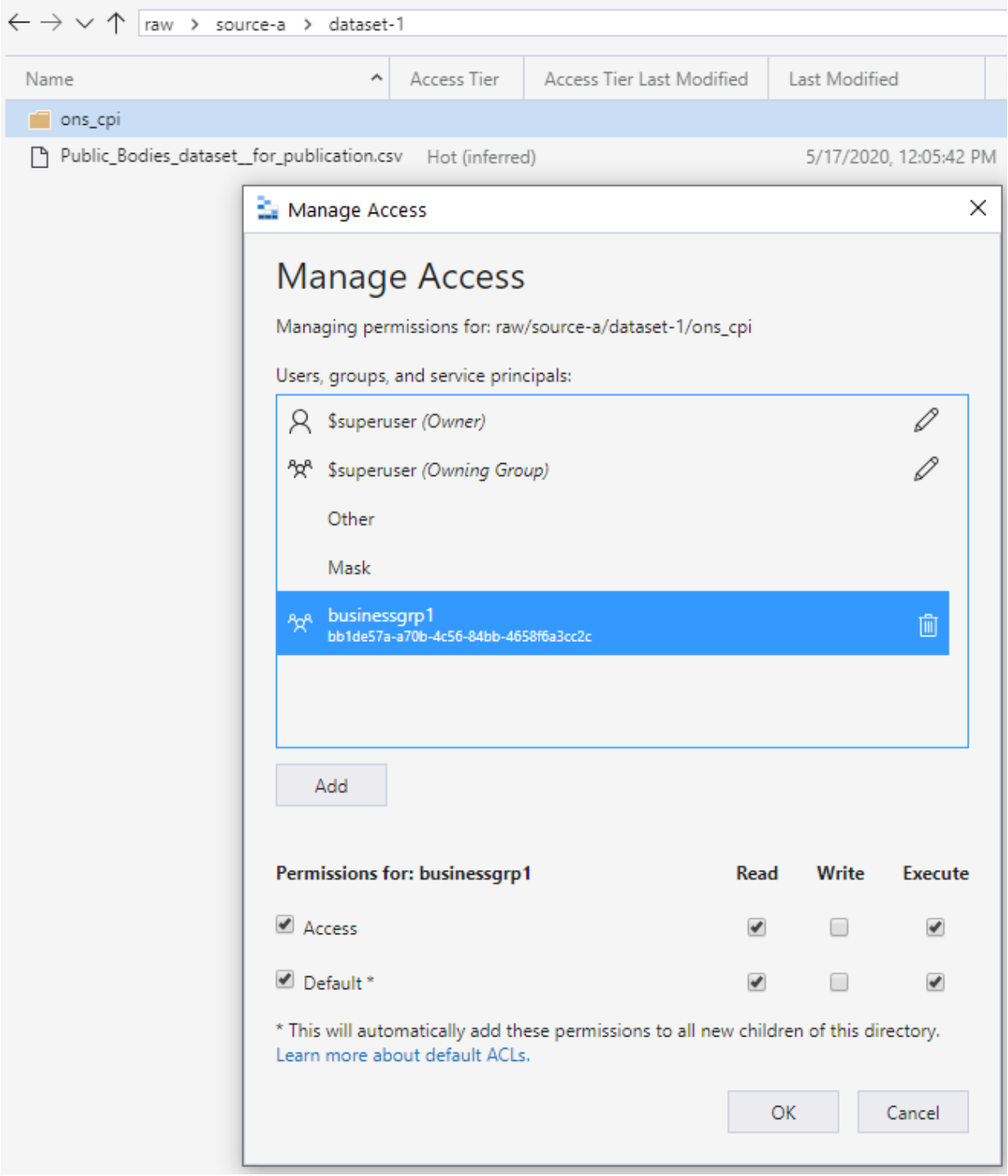


Option 2: The access control list other entry

The recommended approach is to use the ACL other entry set at the container or root. Specify defaults and access ACLs as shown in the following screen. This approach ensures that every part of the path from root to lowest level has run permissions.



This run permission propagates downward to any added child folders. The permission propagates to the depth where the intended access group needs permissions to read and run. The level is in the lowest part of the chain, as shown in the following screen. This approach grants group access to read the data. The approach works similarly for write access.



Recommended data lake zones security

The following usages are the recommended security patterns for each of the data lake zones:

- Raw should allow access to data only by using security principal names (SPNs).
- Enriched should allow access to data only by using security principal names (SPNs).
- Curated should allow access with both security principal names (SPNs) and user principal names (UPNs).

Example scenario for using Azure AD security groups

There are many different ways to set up groups. For example, imagine that you have a directory named `/LogData` that holds log data that's generated by your server. Azure Data Factory ingests data into that folder. Specific users from the service engineering team upload logs and manage other users of this folder. The Azure Databricks analytics and data science workspace clusters could analyze logs from that folder.

To enable these activities, create a `LogsWriter` group and a `LogsReader` group. Assign the following permissions:

- Add the `LogsWriter` group to the ACL of the `/LogData` directory with `rwx` permissions.
- Add the `LogsReader` group to the ACL of the `/LogData` directory with `r-x` permissions.
- Add the service principal object or managed service identity (MSI) for Data Factory to the `LogsWriters` group.
- Add users in the service engineering team to the `LogsWriter` group.
- Azure Databricks is configured for Azure AD passthrough to the Azure Data Lake store.

If a user in the service engineering team transfers to a different team, just remove that user from the `LogsWriter` group.

If you didn't add that user to a group, but instead, added a dedicated ACL entry for that user, you'll need to remove that ACL entry from the `/LogData` directory. You'll also need to remove the entry from all subdirectories and files in the entire directory hierarchy of the `/LogData` directory.

Azure Synapse Analytics data access control

To deploy an Azure Synapse workspace, it requires an Azure Data Lake Storage Gen2 account. Azure Synapse Analytics uses the primary storage account for several integration scenarios and stores data in a container. The container includes Apache Spark tables and application logs under a folder called `/synapse/{workspaceName}`. The workspace also uses a container for managing libraries that you install.

During the workspace deployment through the Azure portal, provide an existing storage account or create a new one. The provided storage account is the primary storage account for the workspace. The deployment process grants the workspace identity

access to the specified Data Lake Storage Gen2 account, using the **Storage Blob Data Contributor** role.

If you deploy the workspace outside of the Azure portal, manually add Azure Synapse Analytics workspace identity to the **Storage Blob Data Contributor** role. It's recommended you assign the role **Storage Blob Data Contributor** on the container level to follow the least privilege principle.

When you run pipelines, workflows, and notebooks through jobs, they use the workspace identity permission context. If any of the jobs read or write to the workspace primary storage, the workspace identity uses the read/write permissions granted through the **Storage Blog Data Contributor**.

When users sign in to the workspace to run scripts or for development, the user's context permissions allow read/write access on the primary storage.

Azure Synapse Analytics fine-grained data access control using access control lists

When you set up data lake access control, some organizations require granular level access. They might have sensitive data that can't be seen by some groups in the organization. Azure RBAC allows read or write at the storage account and container level only. With ACLs, you can set up fine-grained access control at the folder and file level to allow read/write on a subset of data for specific groups.

Considerations when using Spark tables

When you use Apache Spark tables in Spark pool, it creates a warehouse folder. The folder is in the root of the container in the workspace primary storage:

Output

```
synapse/workspaces/{workspaceName}/warehouse
```

If you plan to create Apache Spark tables in Azure Synapse Spark pool, grant write permission on the **warehouse** folder for the group running the command that creates the Spark table. If the command runs through a triggered job in a pipeline, grant write permission to the workspace MSI.

This example creates a Spark table:

Python

```
df.write.saveAsTable("table01")
```

For more information, see How to set up access control for your synapse workspace.

Summary of Azure Data Lake access

No single approach to managing data lake access suits everyone. A major benefit of a data lake is to provide friction-free access to data. In practice, different organizations want different levels of governance and control over their data. Some organizations have a centralized team to manage access and provision groups under rigorous internal controls. Other organizations are more agile and have decentralized control. Choose the approach that meets your level of governance. Your choice shouldn't result in undue delays or friction in gaining access to data.

Next steps

- Use Azure Databricks within cloud-scale analytics in Azure

When is ADLS Gen2 the right choice for your data lake?

Key considerations in designing your data lake

Terminology

Organizing and managing data in your data lake

Do I want a centralized or a federated data lake implementation?

How do I organize my data?

How do I manage access to my data?

What data format do I choose?

How do I manage my data lake cost?

How do I monitor my data lake?

Optimizing your data lake for better scale and performance

File sizes and number of files

File Formats

Partitioning schemes

Use Query Acceleration

Recommended reading

The Hitchhiker's Guide to the Data Lake

A comprehensive guide on key considerations involved in building your enterprise data lake

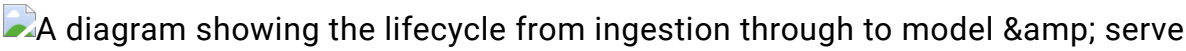
> Share this page using https://aka.ms/adls/hitchhikersguide

- The Hitchhiker's Guide to the Data Lake
  - When is ADLS Gen2 the right choice for your data lake?
  - Key considerations in designing your data lake
  - Terminology
  - Organizing and managing data in your data lake
    - Do I want a centralized or a federated data lake implementation?
    - How do I organize my data?
    - How do I manage access to my data?
    - What data format do I choose?
    - How do I manage my data lake cost?

        - How do I monitor my data lake?
    - Optimizing your data lake for better scale and performance
        - File sizes and number of files
        - File Formats
        - Partitioning schemes
        - Use Query Acceleration
    - Recommended reading

Azure Data Lake Storage Gen2 (ADLS Gen2) is a highly scalable and cost-effective data lake solution for big data analytics. As we continue to work with our customers to unlock key insights out of their data using ADLS Gen2, we have identified a few key patterns and considerations that help them effectively utilize ADLS Gen2 in large scale Big Data platform architectures.

This document captures these considerations and best practices that we have learnt based on working with our customers. For the purposes of this document, we will focus on the Modern Data Warehouse pattern used prolifically by our large-scale enterprise customers on Azure , including our solutions such as Azure Synapse Analytics.



We will improve this document to include more analytics patterns in future iterations.

> Important: Please consider the content of this document as guidance and best practices to help you make your architectural and implementation decisions. This is not an official HOW-TO documentation.

When is ADLS Gen2 the right choice for your data lake?

An enterprise data lake is designed to be a central repository of unstructured , semi-structured and structured data used in your big data platform. The goal of the enterprise data lake is to eliminate data silos (where the data can only be accessed by one part of your organization) and promote a single storage layer that can accommodate the various data needs of the organization For more information on picking the right storage for your solution, please visit the Choosing a big data storage technology in Azure article.

A common question that comes up is when to use a data warehouse vs a data lake. We urge you to think about data lake and data warehouse as complementary solutions that work together to help you derive key insights from your data. A data lake is a store for all types of data from various sources. The data in its natural form is stored as raw data, and schema and transformations are applied on this raw data to gain valuable business insights depending on the key questions the business is trying to answer. A data warehouse is a store

for highly structured schematized data that is usually organized and processed to derive very specific insights. E.g. a retail customer can store the past 5 years' worth of sales data in a data lake, and in addition they can process data from social media to extract the new trends in consumption and intelligence from retail analytics solutions on the competitive landscape and use all these as input together to generate a data set that can be used to project the next years sales targets. They can then store the highly structured data in a data warehouse where BI analysts can build the target sales projections. In addition, they can use the same sales data and social media trends in the data lake to build intelligent machine learning models for personalized recommendations on their website.

ADLS Gen2 is an enterprise ready hyperscale repository of data for your big data analytics workloads. ADLS Gen2 offers faster performance and Hadoop compatible access with the hierarchical namespace, lower cost and security with fine grained access controls and native AAD integration. This lends itself as the choice for your enterprise data lake focused on big data analytics scenarios – extracting high value structured data out of unstructured data using transformations, advanced analytics using machine learning or real time data ingestion and analytics for fast insights. Its worth noting that we have seen customers have different definition of what hyperscale means – this depends on the data stored, the number of transactions and the throughput of the transactions. When we say hyperscale, we are typically referring to multi-petabytes of data and hundreds of Gbps in throughput – the challenges involved with this kind of analytics is very different from a few hundred GB of data and a few Gbps of transactions in throughput.

Key considerations in designing your data lake

As you are building your enterprise data lake on ADLS Gen2, its important to understand your requirements around your key use cases, including

1. What am I storing in my data lake?
2. How much data am I storing in the data lake?
3. What portion of your data do you run your analytics workloads on?
4. Who needs access to what parts of my data lake?
5. What are the various analytics workloads that I'm going to run on my data lake?
6. What are the various transaction patterns on the analytics workloads?
7. What is the budget I'm working with?

We would like to anchor the rest of this document in the following structure for a few key design/architecture questions that we have heard consistently from our customers.

- Available options with the pros and cons
- Factors to consider when picking the option that works for you
- Recommended patterns where applicable

- Anti-patterns that you want to avoid

To best utilize this document, identify your key scenarios and requirements and weigh in our options against your requirements to decide on your approach. If you are not able to pick an option that perfectly fits your scenarios, we recommend that you do a proof of concept (PoC) with a few options to let the data guide your decision.

Terminology

Before we talk about the best practices in building your data lake, it's important to get familiar with the various terminology we will use this document in the context of building your data lake with ADLS Gen2. This document assumes that you have an account in Azure.

Resource: A manageable item that is available through Azure. Virtual machines, storage accounts, VNETs are examples of resources.

Subscription: An Azure subscription is a logical entity that is used to separate the administration and financial (billing) logic of your Azure resources. A subscription is associated with limits and quotas on Azure resources, you can read about them here.

Resource group: A logical container to hold the resources required for an Azure solution can be managed together as a group. You can read more about resource groups here.

Storage account: An Azure resource that contains all of your Azure Storage data objects: blobs, files, queues, tables and disks. You can read more about storage accounts here. For the purposes of this document, we will be focusing on the ADLS Gen2 storage account – which is essentially a Azure Blob Storage account with Hierarchical Namespace enabled, you can read more about it here.

container (also referred to as container for non-HNS enabled accounts): A container organizes a set of objects (or files). A storage account has no limits on the number of containers, and the container can store an unlimited number of folders and files. There are properties that can be applied at a container level such as RBACs and SAS keys.

Folder/Directory: A folder (also referred to as a directory) organizes a set of objects (other folders or files). There are no limits on how many folders or files can be created under a folder. A folder also has access control lists (ACLs) associated with it, there are two types of ACLs associated with a folder – access ACLs and default ACLs, you can read more about them here.

Object/file: A file is an entity that holds data that can be read/written. A file has an access control list associated with it. A file has only access ACLs and no default ACLs.

Organizing and managing data in your data lake

As our enterprise customers build out their data lake strategy, one of the key value proposition of ADLS Gen2 is to serve as the single data store for all their analytics scenarios. Please remember that this single data store is a logical entity that could manifest either as a single ADLS Gen2 account or as multiple accounts depending on the design considerations. Some customers have end to end ownership of the components of an analytics pipeline, and other customers have a central team/ organization managing the infrastructure, operations and governance of the data lake while serving multiple customers – either other organizations in their enterprise or other customers external to their enterprise.

In this section, we have addressed our thoughts and recommendations on the common set of questions that we hear from our customers as they design their enterprise data lake. For illustration, we will take the example of a large retail customer, Contoso.com, building out their data lake strategy to help with various predictive analytics scenarios.

Do I want a centralized or a federated data lake implementation?

As an enterprise data lake, you have two available options – either centralize all the data management for your analytics needs within one organization, or have a federated model, where your customers manage their own data lakes while the centralized data team provides guidance and also manages a few key aspects of the data lake such as security and data governance. It is important to remember that both the centralized and federated data lake strategies can be implemented with one single storage account or multiple storage accounts.

A common question our customers ask us is if they can build their data lake in a single storage account or if they need multiple storage accounts. While technically a single ADLS Gen2 could solve your business needs, there are various reasons why a customer would choose multiple storage accounts, including, but not limited to the following scenarios in the rest of this section.

Key considerations

When deciding the number of storage accounts you want to create, the following considerations are helpful in deciding the number of storage accounts you want to provision.

- A single storage account gives you the ability to manage a single set of control plane management operations such as RBACs, firewall settings, data lifecycle management policies for all the data in your storage account, while allowing you to organize your data using containers, files and folders on the storage account. If you want to optimize for ease of management, specially if you adopt a centralized data lake strategy, this would be a good model to consider.

- Multiple storage accounts provide you the ability to isolate data across different accounts so different management policies can be applied to them or manage their billing/cost logic separately. If you are considering a federated data lake strategy with each organization or business unit having their own set of manageability requirements, then this model might work best for you.

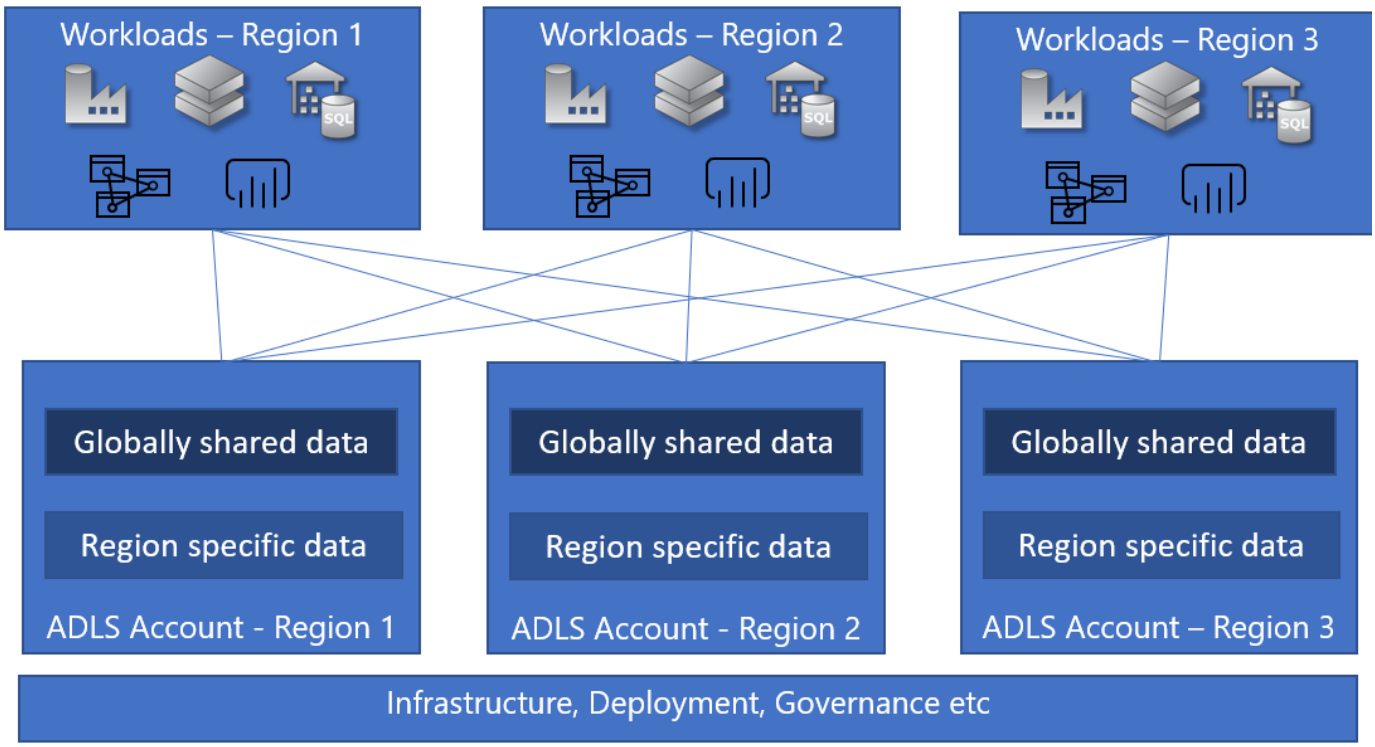
Let us put these aspects in context with a few scenarios.

Enterprise data lake with a global footprint

Driven by global markets and/or geographically distributed organizations, there are scenarios where enterprises have their analytics scenarios factoring multiple geographic regions. The data itself can be categorized into two broad categories.

- Data that can be shared globally across all regions – E.g. Contoso is trying to project their sales targets for the next fiscal year and want to get the sales data from their various regions.
- Data that needs to be isolated to a region – E.g. Contoso wants to provide a personalized buyer experience based on their profile and buying patterns. Given this is customer data, there are sovereignty requirements that need to be met, so the data cannot leave the region.

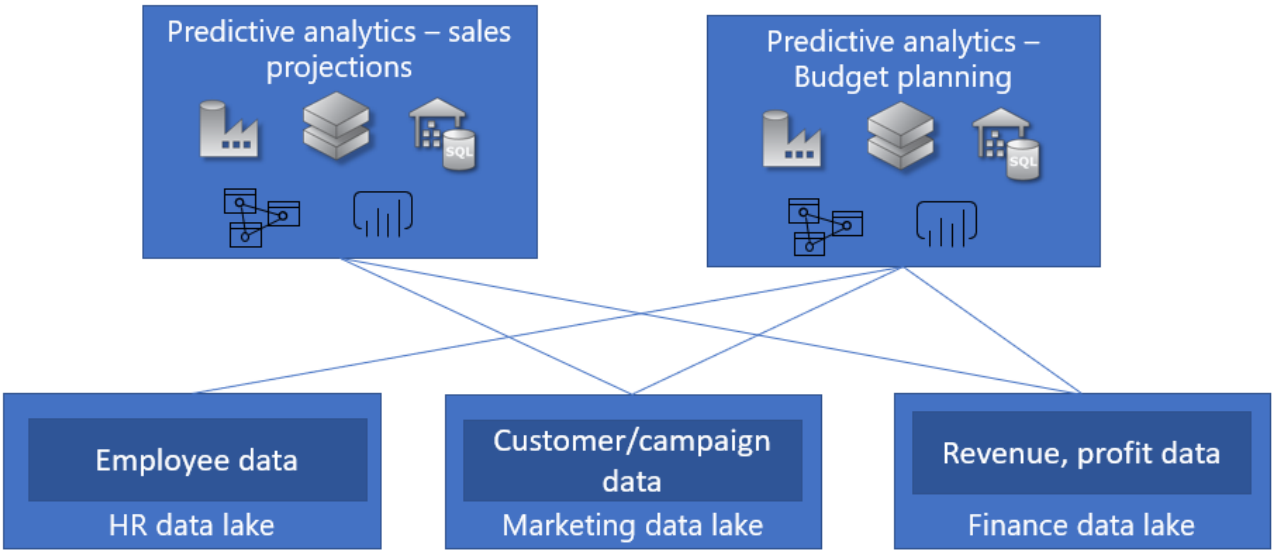
In this scenario, the customer would provision region-specific storage accounts to store data for a particular region and allow sharing of specific data with other regions. There is still one centralized logical data lake with a central set of infrastructure management, data governance and other operations that comprises of multiple storage accounts here.



Customer or data specific isolation

There are scenarios where enterprise data lakes serve multiple customer (internal/external) scenarios that may be subject to different requirements – different query patterns and different access requirements. Let us take our Contoso.com example where they have analytics scenarios to manage the company operations. In this case, they have various data sources – employee data, customers/campaign data and financial data that are subject to different governance and access rules and are also possibly managed by different organizations within the company. In this case, they could choose to create different data lakes for the various data sources.

In another scenario, enterprises that serve as a multi-tenant analytics platform serving multiple customers could end up provisioning individual data lakes for their customers in different subscriptions to help ensure that the customer data and their associated analytics workloads are isolated from other customers to help manage their cost and billing models.



Recommendations

- Create different storage accounts (ideally in different subscriptions) for your development and production environments. In addition to ensuring that there is enough isolation between your development and production environments requiring different SLAs, this also helps you track and optimize your management and billing policies efficiently.
- Identify the different logical sets of your data and think about your needs to manage them in a unified or isolated fashion – this will help determine your account boundaries.
- Start your design approach with one storage account and think about reasons why you need multiple storage accounts (isolation, region based requirements etc) instead of the other way around.
- There are also subscription limits and quotas on other resources (such as VM cores, ADF instances) – factor that into consideration when designing your data lake.

Anti-patterns

Beware of multiple data lake management

When you decide on the number of ADLS Gen2 storage accounts, ensure that you are optimizing for your consumption patterns. If you do not require isolation and you are not utilizing your storage accounts to their fullest capabilities, you will be incurring the overhead of managing multiple accounts without a meaningful return on investment.

Copying data back and forth

When you have multiple data lakes, one thing you would want to treat carefully is if and how you are replicating data across the multiple accounts. This creates a management problem of what is the source of truth and how fresh it needs to be, and also consumes transactions involved in copying data back and forth. We have features in our roadmap that makes this workflow easier if you have a legitimate scenario to replicate your data.

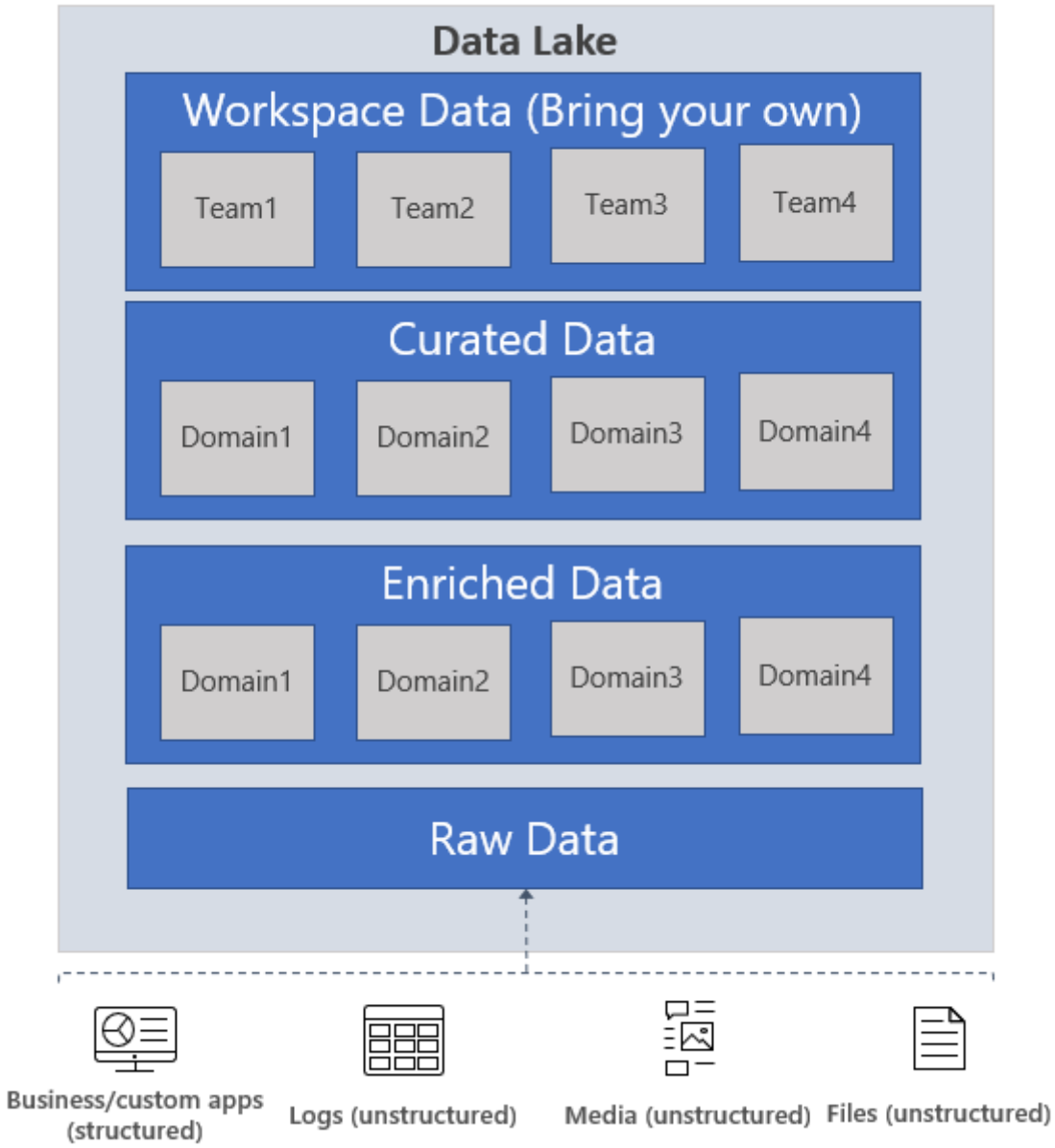
A note on scale

One common question that our customers ask is if a single storage account can infinitely continue to scale to their data, transaction and throughput needs. Our goal in ADLS Gen2 is to meet the customer where they want in terms of their limits. We do request that when you have a scenario where you have requirements for really storing really large amounts of data (multi-petabytes) and require the account to support a really large transaction and throughput pattern (tens of thousands of TPS and hundreds of Gbps throughput), typically observed by requiring 1000s of cores of compute power for analytics processing via Databricks or HDInsight, please do contact our product group so we can plan to support your requirements appropriately.

How do I organize my data?

Data organization in a an ADLS Gen2 account can be done in the hierarchy of containers, folders and files in that order, as we saw above. A very common point of discussion as we work with our customers to build their data lake strategy is how they can best organize their data. There are multiple approaches to organizing the data in a data lake, this section documents a common approach that has been adopted by many customers building a data platform.

This organization follows the lifecycle of the data as it flows through the source systems all the way to the end consumers – the BI analysts or Data Scientists. As an example, let us follow the journey of sales data as it travels through the data analytics platform of Contoso.com.



As an example , think of the raw data as a lake/pond with water in its natural state, the data is ingested and stored as is without transformations, the enriched data is water in a reservoir that is cleaned and stored in a predictable state (schematized in the case of our data), the curated data is like bottled water that is ready for consumption. Workspace data is like a laboratory where scientists can bring their own for testing. Its worth noting that while all this data layers are present in a single logical data lake, they could be spread across different physical storage accounts. In these cases, having a metastore is helpful for discovery.

- **Raw data**: This is data as it comes from the source systems. This data is stored as is in the data lake and is consumed by an analytics engine such as Spark to perform cleansing and enrichment operations to generate the curated data. The data in the raw zone is sometimes also stored as an aggregated data set, e.g. in the case of streaming scenarios, data is ingested via message bus such as Event Hub, and then aggregated via a real time processing engine such as Azure Stream analytics or Spark Streaming before storing in the data lake. Depending on what your business needs, you can choose to leave the data as is (E.g. log messages from servers) or aggregate it (E.g. real time

streaming data). This layer of data is highly controlled by the central data engineering team and is rarely given access to other consumers. Depending on the retention policies of your enterprise, this data is either stored as is for the period required by the retention policy or it can be deleted when you think the data is of no more use. E.g. this would be raw sales data that is ingested from Contoso's sales management tool that is running in their on-prem systems.

- **Enriched data**: This layer of data is the version where raw data (as is or aggregated) has a defined schema and also, the data is cleansed, enriched (with other sources) and is available to analytics engines to extract high value data. Data engineers generate these datasets and also proceed to extract high value/curated data from these datasets. E.g. this would be enriched sales data – ensuring that the sales data is schematized, enriched with other product or inventory information and also separated into multiple datasets for the different business units inside Contoso.
- **Curated data**: This layer of data contains the high value information that is served to the consumers of the data – the BI analysts and the data scientists. This data has structure and can be served to the consumers either as is (E.g. data science notebooks) or through a data warehouse. Data assets in this layer is usually highly governed and well documented. E.g. high-quality sales data (that is data in the enriched data zone correlated with other demand forecasting signals such as social media trending patterns) for a business unit that is used for predictive analytics on determining the sales projections for the next fiscal year.
- **Workspace data**: In addition to the data that is ingested by the data engineering team from the source, the consumers of the data can also choose to bring other data sets that could be valuable. In this case, the data platform can allocate a workspace for these consumers so they can use the curated data along with the other data sets they bring to generate valuable insights. E.g. a Data Science team is trying to determine the product placement strategy for a new region, they could bring other data sets such as customer demographics and data on usage of other similar products from that region and use the high value sales insights data to analyze the product market fit and the offering strategy.
- **Archive data**: This is your organization's data 'vault' - that has data stored to primarily comply with retention policies and has very restrictive usage, such as supporting audits. You can use the Cool and Archive tiers in ADLS Gen2 to store this data. You can read more about our data lifecycle management policies to identify a plan that works for you.

Key considerations

When deciding the structure of your data, consider both the semantics of the data itself as well as the consumers who access the data to identify the right data organization strategy for you.

Recommendations

- Create different folders or containers (more below on considerations between folders vs containers) for the different data zones - raw, enriched, curated and workspace data sets.
- Inside a zone, choose to organize data in folders according to logical separation, e.g. datetime or business units or both. You can find more examples and scenarios on directory layout in our best practices document.
    - Consider the analytics consumption patterns when designing your folder structures. E.g. if you have a Spark job reading all sales data of a product from a specific region for the past 3 months, then an ideal folder structure here would be /enriched/product/region/timestamp.
    - Consider the access control model you would want to follow when deciding your folder structures.
    - The table below provides a framework for you to think about the different zones of the data and the associated management of the zones with a commonly observed pattern.

Consideration	Raw data	Enriched data	Curated data
Consumer	Data engineering team	Data engineering team, with adhoc access patterns by the Data scientists/BI analysts	Data enginee analysts, Dat
Access control	Locked for access by data engineering team	Full control to data engineering team, with read access to the BI analysts/data scientists	Full control t engineering t read and writ the BI analys scientists
Data lifecycle management	Once enriched data is generated, can be moved to a cooler tier of storage to manage costs.	Older data can be moved to a cooler tier.	Older data ca to a cooler ti
Folder structure and hierarchy	Folder structure to mirror the ingestion patterns.	Folder structure mirrors organization, e.g. business units.	Folder struct organization, units.
Example	`/raw/sensordata` `/raw/lobappdata`	`/enriched/sales` `/enriched/manufacturing`	`/curated/s` `/curated/m`

Consideration	Raw data	Enriched data	Curated data
	`/raw/userclickdata`		

- Another common questions that our customers ask if when to use containers and when to use folders to organize the data. While at a higher level, they both are used for logical organizations of the data, they have a few key differences.

Consideration	container	Folder
Hierarchy	container can contain folders or files.	Folder can contain other folders or files.
Access control using AAD	At the container level, you can set coarse grained access controls using RBACs. These RBACs apply to all data inside the container.	At the folder level, you can set fine grained access controls using ACLs. The ACLs apply to the folder only (unless you use default ACLs, in which case, a snapshot is taken when new files/folders are created under the folder).
Non-AAD access control	At a container level, you can enable anonymous access (via shared keys) or set SAS keys specific to the container.	A folder does not support non-AAD access control.

Anti-patterns

Indefinite growth of irrelevant data

While ADLS Gen2 storage is not very expensive and lets you store a large amount of data in your storage accounts, lack of lifecycle management policies could end up growing the data in the storage very quickly even if you don't require the entire corpus of data for your scenarios. Two common patterns where we see this kind of data growth is:

- **Data refresh with a newer version of data** – Customers typically keep a few older versions of the data for analysis when there is a period refresh of the same data, e.g. when customer engagement data over the last month is refreshed daily over a rolling window of 30 days, you get 30 days engagement data everyday and when you don't have a clean up process in place, your data could grow exponentially.
- **Workspace data accumulation** – In the workspace data zone, the customers of your data platform, i.e. the BI analysts or data scientists can bring their own data sets

Typically, we have seen that this data could also accumulate just as easily when the data not used is left lying around in the storage spaces.

How do I manage access to my data?

ADLS Gen2 supports access control models that combine both RBACs and ACLs to manage access to the data. You can find more information about the access control here. In addition to managing access using AAD identities using RBACs and ACLs, ADLS Gen2 also supports using SAS tokens and shared keys for managing access to data in your Gen2 account.

A common question that we hear from our customers is when to use RBACs and when to use ACLs to manage access to the data. RBACs let you assign roles to security principals (user, group, service principal or managed identity in AAD) and these roles are associated with sets of permissions to the data in your container. RBACs can help manage roles related to control plane operations (such as adding other users and assigning roles, manage encryption settings, firewall rules etc) or for data plane operations (such as creating containers, reading and writing data etc). Fore more information on RBACs, you can read this article.

RBACs are essentially scoped to top-level resources – either storage accounts or containers in ADLS Gen2. You can also apply RBACs across resources at a resource group or subscription level. ACLs let you manage a specific set of permissions for a security principal to a much narrower scope – a file or a directory in ADLS Gen2. There are 2 types of ACLs – Access ADLs that control access to a file or a directory, Default ACLs are templates of ACLs set for directories that are associated with a directory, a snapshot of these ACLs are inherited by any child items that are created under that directory.

Key considerations

The table below provides a quick overview of how ACLs and RBACs can be used to manage permissions to the data in your ADLS Gen2 accounts – at a high level, use RBACs to manage coarse grained permissions (that apply to storage accounts or containers) and use ACLs to manage fine grained permissions (that apply to files and directories).

Consideration	RBACs	ACLs
Scope	Storage accounts, containers. Cross resource RBACs at subscription or resource group level.	Files, directories
Limits	2000 RBACs in a subscription	32 ACLs (effectively 28 ACLs) per file, 32 ACLs (effectively 28

Consideration	RBACs	ACLs
		ACLs) per folder, default and access ACLs each
Supported levels of permission	Built-in RBACs or custom RBACs	ACL permissions

When using RBAC at the container level as the only mechanism for data access control, be cautious of the 2000 limit, particularly if you are likely to have a large number of containers. You can view the number of role assignments per subscription in any of the access control (IAM) blades in the portal.

Recommendations

- Create security groups for the level of permissions you want for an object (typically a directory from what we have seen with our customers) and add them to the ACLs. For specific security principals you want to provide permissions, add them to the security group instead of creating specific ACLs for them. Following this practice will help you minimize the process of managing access for new identities – which would take a really long time if you want to add the new identity to every single file and folder in your container recursively. Let us take an example where you have a directory, /logs, in your data lake with log data from your server. You ingest data into this folder via ADF and also let specific users from the service engineering team upload logs and manage other users to this folder. In addition, you also have various Databricks clusters analyzing the logs. You will create the /logs directory and create two AAD groups LogsWriter and LogsReader with the following permissions.
  - LogsWriter added to the ACLs of the /logs folder with rwx permissions.
  - LogsReader added to the ACLs of the /logs folder with r-x permissions.
  - The SPNs/MSIs for ADF as well as the users and the service engineering team can be added to the LogsWriter group.
  - The SPNs/MSIs for Databricks will be added to the LogsReader group.

What data format do I choose?

Data may arrive to your data lake account in a variety of formats – human readable formats such as JSON, CSV or XML files, compressed binary formats such as .tar.gz and a variety of sizes – huge files (a few TBs) such as an export of a SQL table from your on-premise systems or a large number of tiny files (a few KBs) such as real-time events from your IoT solution. While ADLS Gen2 supports storing all kinds of data without imposing any restrictions, it is better to think about data formats to maximize efficiency of your processing pipelines and optimize costs – you can achieve both of these by picking the right format and the right file sizes.

Hadoop has a set of file formats it supports for optimized storage and processing of structured data. Let us look at some common file formats – Avro, Parquet and ORC. All of these are machine-readable binary file formats, offer compression to manage the file size and are self-describing in nature with a schema embedded in the file. The difference between the formats is in how data is stored – Avro stores data in a row-based format and Parquet and ORC formats store data in a columnar format.

Key considerations

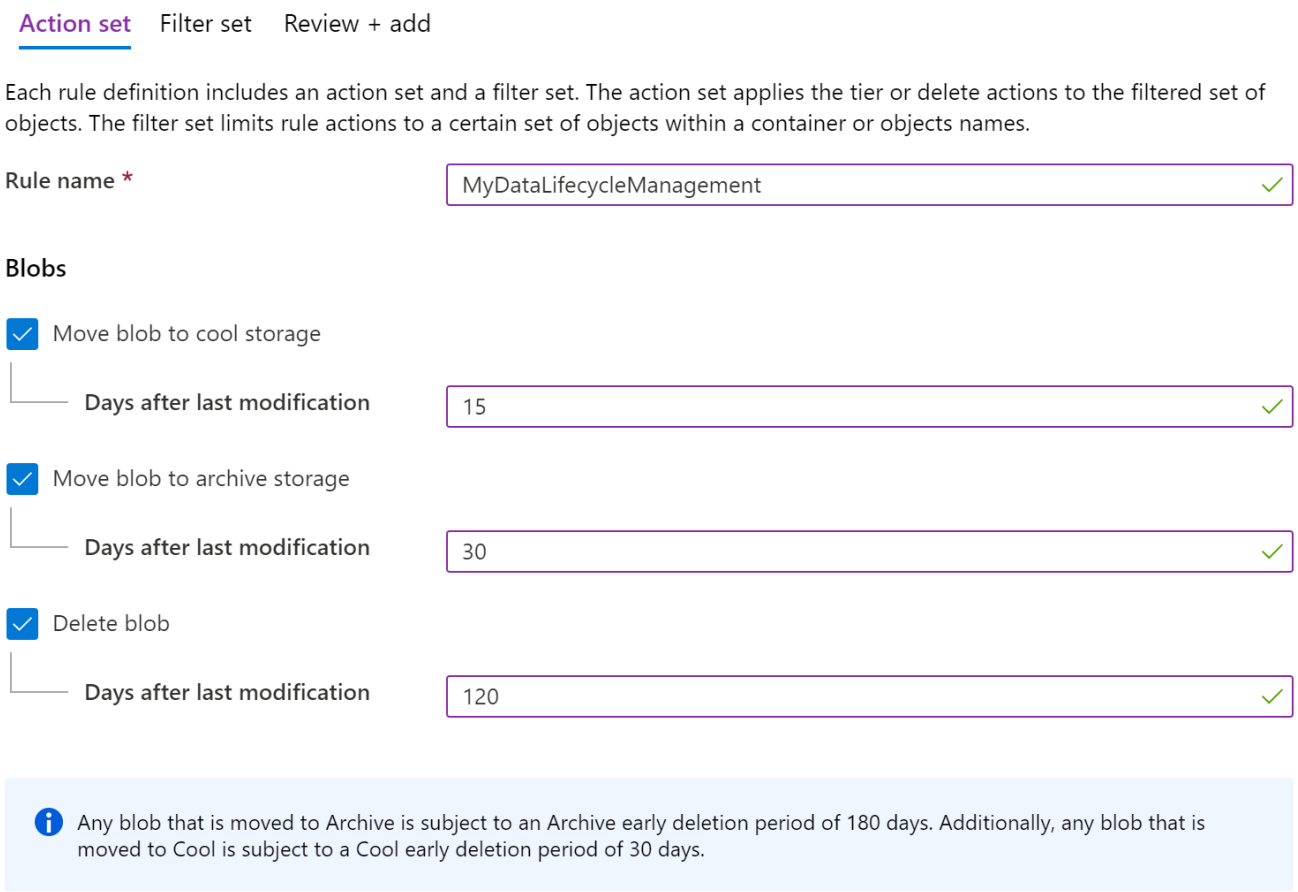
- Avro file format is favored where the I/O patterns are more write heavy or the query patterns favor retrieving multiple rows of records in their entirety. E.g. Avro format is favored by message bus such as Event Hub or Kafka writes multiple events/messages in succession.
- Parquet and ORC file formats are favored when the I/O patterns are more read heavy and/or when the query patterns are focused on a subset of columns in the records – where the read transactions can be optimized to retrieve specific columns instead of reading the entire record.

How do I manage my data lake cost?

ADLS Gen2 offers a data lake store for your analytics scenarios with the goal of lowering your total cost of ownership. The pricing for ADLS Gen2 can be found here. As our enterprise customers serve the needs of multiple organizations including analytics use-cases on a central data lake, their data and transactions tend to increase dramatically. With little or no centralized control, so will the associated costs increase. This section provides key considerations that you can use to manage and optimize the cost of your data lake.

Key considerations

- ADLS Gen2 provides policy management that you can use to leverage the lifecycle of data stored in your Gen2 account. You can read more about these policies here. E.g. if your organization has a retention policy requirement to keep the data for 5 years, you can set a policy to automatically delete the data if it has not been modified for 5 years. If your analytics scenarios primarily operate on data that is ingested in the past month, you can move the data older than the month to a lower tier (cool or archive) which have a lower cost for data stored. Please note that the lower tiers have a lower price for data at rest, but higher policies for transactions, so do not move data to lower tiers if you expect the data to be frequently transacted on.



- Ensure that you are choosing the right replication option for your accounts, you can read the data redundancy article to learn more about your options. E.g. while GRS accounts ensure that your data is replicated across multiple regions, it also costs higher than an LRS account (where data is replicated on the same datacenter). When you have a production environment, replication options such as GRS are highly valuable to ensure business continuity with high availability and disaster recovery. However, an LRS account might just suffice for your development environment.
- As you can see from the pricing page of ADLS Gen2, your read and write transactions are billed in 4 MB increments. E.g. if you do 10,000 read operations and each file read is 16 MB in size, you will be charged for 40,000 transactions. When you have scenarios where you read a few KBs of data in a transaction, you will still be charged for the a 4 MB transaction. Optimizing for more data in a single transaction, i.e. optimizing for higher throughtput in your transactions does not just save cost, but also highly improves your performance.

How do I monitor my data lake?

Understanding how your data lake is used and how it performs is a key component of operationalizing your service and ensuring it is available for use by any workloads which consume the data contained within it. This includes:

- Being able to audit your data lake in terms of frequent operations

- Having visibility into key performance indicators such as operations with high latency
- Understanding common errors, the operations that caused the error, and operations which cause service-side throttling

Key considerations

All of the telemetry for your data lake is available through Azure Storage logs in Azure Monitor. Azure Storage logs in Azure Monitor is a new preview feature for Azure Storage which allows for a direct integration between your storage accounts and Log Analytics, Event Hubs, and archival of logs to another storage account utilizing standard diagnostic settings. A reference of the the full list of metrics and resources logs and their associated schema can be found in the Azure Storage monitoring data reference.

- Where your choose to store your logs from Azure Storage logs becomes important when you consider how you will access it:
  - If you want to access your logs in near real-time and be able to correlate events in logs with other metrics from Azure Monitor, you can store your logs in a Log Analytics workspace. This allows you to query your logs using KQL and author queries which enumerate the `StorageBlobLogs` table in your workspace.
  - If you want to store your logs for both near real-time query and long term retention, you can configure your diagnostic settings to send logs to both a Log Analytics workspace and a storage account.
  - If you want to access your logs through another query engine such as Splunk, you can configure your dianostic settings to send logs to an Event Hub and ingest logs from the Event Hub to your chosen destination.
- Azure Storage logs in Azure Monitor can be enabled through the Azure Portal, PowerShell, the Azure CLI, and Azure Resource Manager templates. For at-scale deployments, Azure Policy can be used with full support for remediation tasks. For more details, see:
  - Azure/Community-Policy
  - ciphertxt/AzureStoragePolicy

Common KQL queries for Azure Storage logs in Azure Monitor

The following queries can be used to discover insights into the performance and health of your data lake:

- **Frequent operations**

```
StorageBlobLogs
| where TimeGenerated > ago(3d)
| summarize count() by OperationName
| sort by count_ desc
```

```
| render piechart
```

- **High latency operations**

```
StorageBlobLogs
| where TimeGenerated > ago(3d)
| top 10 by DurationMs desc
| project TimeGenerated, OperationName, DurationMs, ServerLatencyMs, Clie
```

- **Operations causing the most errors**

```
  StorageBlobLogs
| where TimeGenerated > ago(3d) and StatusText !contains "Success"
| summarize count() by OperationName
| top 10 by count_ desc
```

A list of all of the built-in queries for Azure Storage logs in Azure Monitor is available in the Azure Montior Community on GitHub in the Azure Services/Storage accounts/Queries folder.

Optimizing your data lake for better scale and performance

> **Under construction, looking for contributions**

In this section, we will address how to optimize your data lake store for your performance in your analytics pipeline. in this section, we will focus on the basic principles that help you optimize the storage transactions. To ensure we have the right context, there is no silver bullet or a 12 step process to optimize your data lake since a lot of considerations depend on the specific usage and the business problems you are trying to solve. However, when we talk about optimizing your data lake for performance, scalability and even cost, it boils down to two key factors:

- Optimize for high throughput – target getting at least a few MBs (higher the better) per transaction.
- Optimize data access patterns – reduce unnecessary scanning of files, read only the data you need to read.

As a pre-requisite to optimizations, it is important for you to understand more about the transaction profile and data organization. Given the varied nature of analytics scenarios, the optimizations depend on your analytics pipeline, storage I/O patterns and the data sets you operate on, specifically the following aspects of your data lake.

> Please note that the scenarios that we talk about is primarily with the focus of optimizing ADLS Gen2 performance. The overall performance of your analytics pipeline would have considerations specific to the analytics engines in addition to the storage performance consideration, our partnerships with the analytics offerings on Azure such as Azure Synapse Analytics, HDInsight and Azure Databricks ensure that we focus on making the overall experience better. In the meantime, while we call out specific engines as examples, please do note that these samples talk primarily about storage performance.

File sizes and number of files

Analytics engines (your ingest or data processing pipelines) incur an overhead for every file they read (related to listing, checking access and other metadata operations) and too many small files can negatively affect the performance of your overall job. Further, when you have files that are too small (in the KBs range), the amount of throughput you achieve with the I/O operations is also low, requiring more I/Os to get the data you want. In general, its a best practice to organize your data into larger sized files (target at least 100 MB or more) for better performance.

In a lot of cases, if your raw data (from various sources) itself is not large, you have the following options to ensure the data set your analytics engines operate on is still optimized with large file sizes.

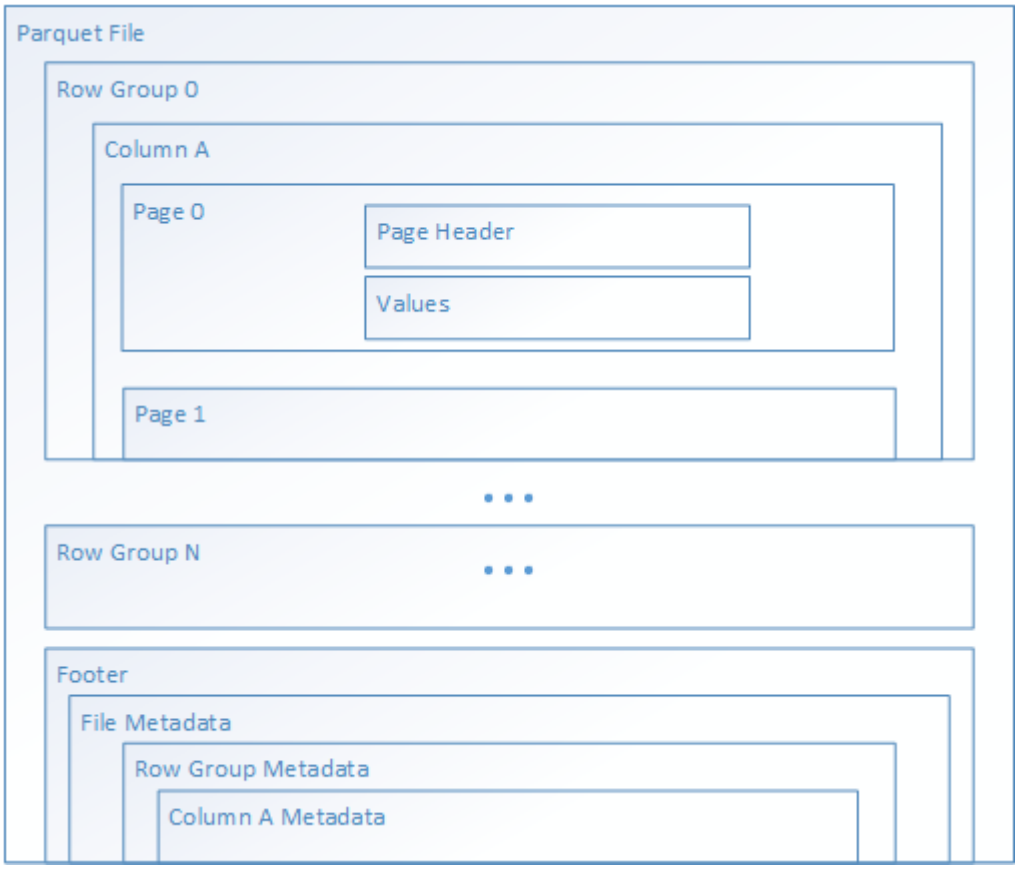
- Add a data processing layer in your analytics pipeline to coalesce data from multiple small files into a large file. You can also use this opportunity to store data in a read-optimized format such as Parquet for downstream processing.
- In the case of processing real time data, you can use a real time streaming engine (such as Azure Stream Analytics or Spark Streaming) in conjunction with a message broker (such as Event Hub or Apache Kafka) to store your data as larger files.

File Formats

As we have already talked about, optimizing your storage I/O patterns can largely benefit the overall performance of your analytics pipeline. It is worth calling out that choosing the right file format can lower your data storage costs in addition to offering better performance. Parquet is one such prevalent data format that is worth exploring for your big data analytics pipeline.

Apache Parquet is an open source file format that is optimized for read heavy analytics pipelines. The columnar storage structure of Parquet lets you skip over non-relevant data making your queries much more efficient. This ability to skip also results in only the data you want being sent from the storage to the analytics engine resulting in lower cost along with

better performance. In addition, since the similar data types (for a column) are stored together, Parquet lends itself friendly to efficient data compression and encoding schemes lowering your data storage costs as well, compared to storing the same data in a text file format.



Services such as Azure Synapse Analytics, Azure Databricks and Azure Data Factory have native functionality built in to take advantage of Parquet file formats as well.

Partitioning schemes

An effective partitioning scheme for your data can improve the performance of your analytics pipeline and also reduce the overall transaction costs incurred with your query. In simplistic terms, partitioning is a way of organizing your data by grouping datasets with similar attributes together in a storage entity, such as a folder. When your data processing pipeline is querying for data with that similar attribute (E.g. all the data in the past 12 hours), the partitioning scheme (in this case, done by datetime) lets you skip over the irrelevant data and only seek the data that you want.

Let us take an example of an IoT scenario at Contoso where data is ingested real time from various sensors into the data lake. Now, you have various options of storing the data, including (but not limited to) the ones listed below:

- Option 1 - `/<sensorid>/<datetime>/<temperature>, <sensorid>/<datetime>/<pressure>, <sensorid>/<datetime>/<humidity>`

- Option 2 - `/<datetime>/<sensorid>/<temperature>, /<datetime>/<sensorid>/<pressure>, /datetime>/<sensorid>/<humidity>`
- Option 3 - `<temperature>/<datetime>/<sensorid>, <pressure>/<datetime>/<sensorid>, <humidity>/<datetime>/<sensorid>`

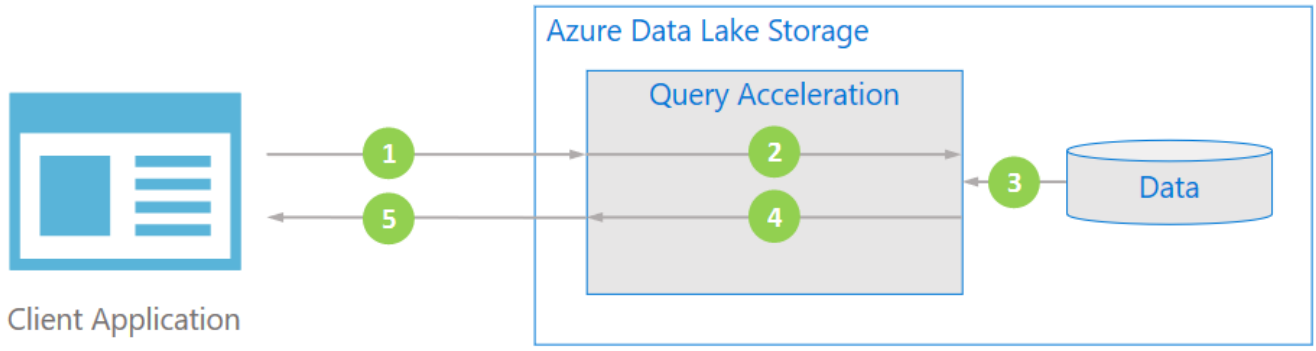
If a high priority scenario is to understand the health of the sensors based on the values they send to ensure the sensors are working fine, then you would have analytics pipelines running every hour or so to triangulate data from a specific sensor with data from other sensors to ensure they are working fine. In this case, Option 2 would be the optimal way for organizing the data.

If instead your high priority scenario is to understand the weather patterns in the area based on the sensor data to ensure what remedial action you need to take, you would have analytics pipelines running periodically to assess the weather based on the sensor data from the area. In this case, you would want to optimize for the organization by date and attribute over the sensorID.

Open source computing frameworks such as Apache Spark provide native support for partitioning schemes that you can leverage in your big data application.

Use Query Acceleration

Azure Data Lake Storage has a capability called Query Acceleration available in preview that is intended to optimize your performance while lowering the cost. Query acceleration lets you filter for the specific rows and columns of data that you want in your dataset by specifying one more predicates (think of these as similar to the conditions you would provide in your WHERE clause in a SQL query) and column projections (think of these as columns you would specify in the SELECT statement in your SQL query) on unstructured data.



In addition to improving performance by filtering the specific data used by the query, Query Acceleration also lowers the overall cost of your analytics pipeline by optimizing the data transferred, and hence reducing the overall storage transaction costs, and also saving you

the cost of compute resources you would have otherwise spun up to read the entire dataset and filter for the subset of data that you need.

Recommended reading

Azure Databricks – Best Practices

Use Azure Data Factory to migrate data from an on-premises Hadoop cluster to ADLS Gen2(Azure Storage)

Use Azure Data Factory to migrate data from an AWS S3 to ADLS Gen2(Azure Storage)

Securing access to ADLS Gen2 from Azure Databricks

Understanding access control and data lake configurations in ADLS Gen2

Access control model in Azure Data Lake Storage Gen2

Article • 05/12/2023

Data Lake Storage Gen2 supports the following authorization mechanisms:

- Shared Key authorization
- Shared access signature (SAS) authorization
- Role-based access control (Azure RBAC)
- Attribute-based access control (Azure ABAC)
- Access control lists (ACL)

Shared Key and SAS authorization grants access to a user (or application) without requiring them to have an identity in Azure Active Directory (Azure AD). With these two forms of authentication, Azure RBAC, Azure ABAC, and ACLs have no effect.

Azure RBAC and ACL both require the user (or application) to have an identity in Azure AD. Azure RBAC lets you grant "coarse-grain" access to storage account data, such as read or write access to **all** of the data in a storage account. Azure ABAC allows you to refine RBAC role assignments by adding conditions. For example, you can grant read or write access to all data objects in a storage account that have a specific tag. ACLs let you grant "fine-grained" access, such as write access to a specific directory or file.

This article focuses on Azure RBAC, ABAC, and ACLs, and how the system evaluates them together to make authorization decisions for storage account resources.

Role-based access control (Azure RBAC)

Azure RBAC uses role assignments to apply sets of permissions to security principals. A security principal is an object that represents a user, group, service principal, or managed identity that is defined in Azure Active Directory (AD). A permission set can give a security principal a "coarse-grain" level of access such as read or write access to **all** of the data in a storage account or **all** of the data in a container.

The following roles permit a security principal to access data in a storage account.

Role	Description
Storage Blob Data Owner	Full access to Blob storage containers and data. This access permits the security principal to set the owner an item, and to modify the ACLs of all items.

Role	Description
Storage Blob Data Contributor	Read, write, and delete access to Blob storage containers and blobs. This access does not permit the security principal to set the ownership of an item, but it can modify the ACL of items that are owned by the security principal.
Storage Blob Data Reader	Read and list Blob storage containers and blobs.

Roles such as Owner, Contributor, Reader, and Storage Account Contributor permit a security principal to manage a storage account, but do not provide access to the data within that account. However, these roles (excluding **Reader**) can obtain access to the storage keys, which can be used in various client tools to access the data.

Attribute-based access control (Azure ABAC)

Azure ABAC builds on Azure RBAC by adding role assignment conditions based on attributes in the context of specific actions. A role assignment condition is an additional check that you can optionally add to your role assignment to provide more refined access control. You cannot explicitly deny access to specific resources using conditions.

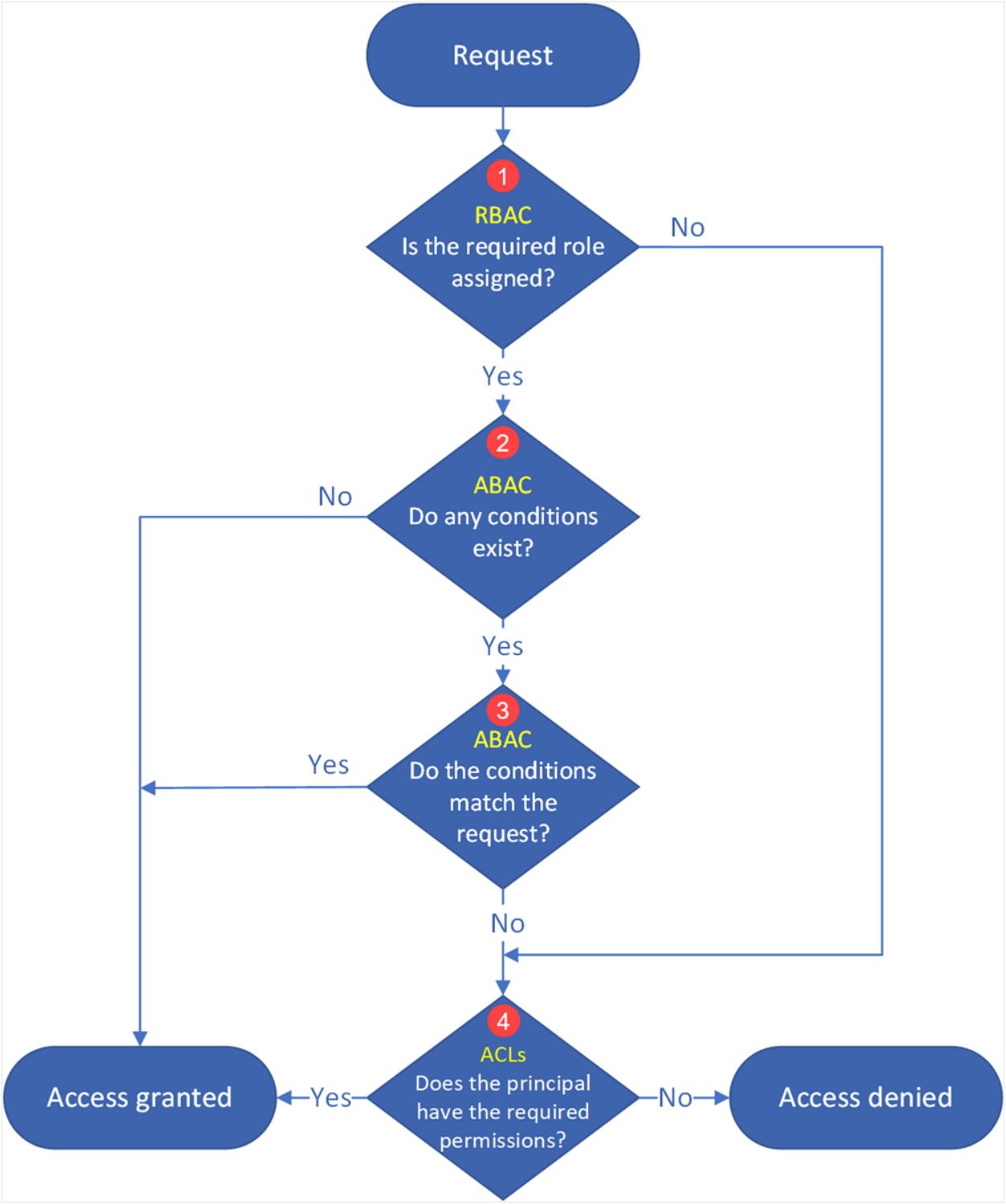
For more information on using Azure ABAC to control access to Azure Storage, see Authorize access to Azure Blob Storage using Azure role assignment conditions.

Access control lists (ACLs)

ACLs give you the ability to apply "finer grain" level of access to directories and files. An *ACL* is a permission construct that contains a series of *ACL entries*. Each ACL entry associates security principal with an access level. To learn more, see Access control lists (ACLs) in Azure Data Lake Storage Gen2.

How permissions are evaluated

During security principal-based authorization, permissions are evaluated as shown in the following diagram.



1. Azure determines whether a role assignment exists for the principal.

    - If a role assignment exists, the role assignment conditions (2) are evaluated next.
    - If not, the ACLs (4) are evaluated next.

2. Azure determines whether any ABAC role assignment conditions exist.

    - If no conditions exist, access is granted.
    - If conditions exist, they are evaluated to see if they match the request (3).

3. Azure determines whether all of the ABAC role assignment conditions match the attributes of the request.

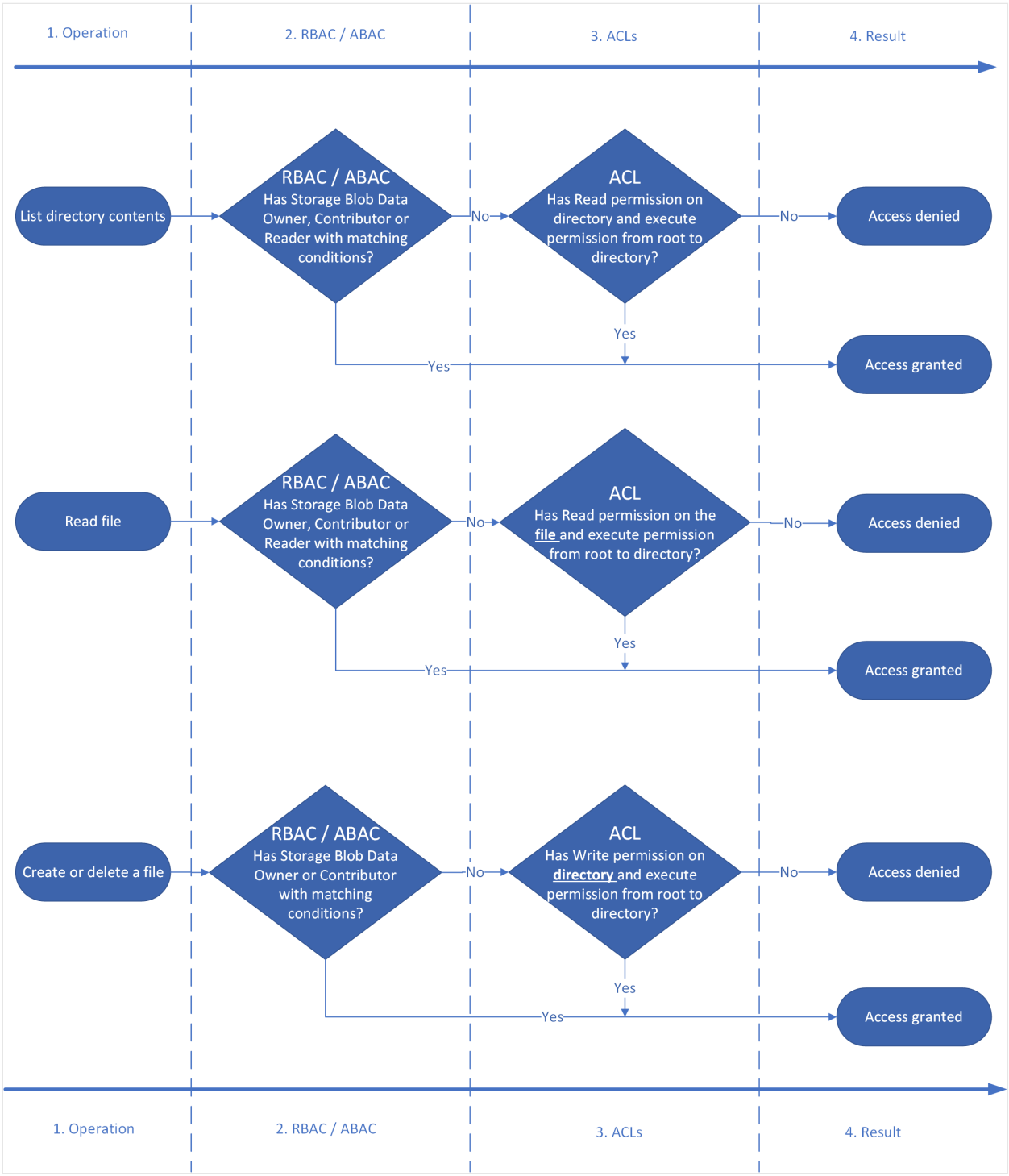
    - If all of them match, access is granted.
    - If at least one of them does not match, the ACLs (4) are evaluated next.

4. If access has not been explicitly granted after evaluating the role assignments and conditions, the ACLs are evaluated.

    - If the ACLs permit the requested level of access, access is granted.
    - If not, access is denied.

> **Important**
>
> Because of the way that access permissions are evaluated by the system, you **cannot** use an ACL to **restrict** access that has already been granted by a role assignment and its conditions. That's because the system evaluates Azure role assignments and conditions first, and if the assignment grants sufficient access permission, ACLs are ignored.

The following diagram shows the permission flow for three common operations: listing directory contents, reading a file, and writing a file.



Permissions table: Combining Azure RBAC, ABAC, and ACLs

The following table shows you how to combine Azure roles, conditions, and ACL entries so that a security principal can perform the operations listed in the **Operation** column. This table shows a column that represents each level of a fictitious directory hierarchy. There's a column for the root directory of the container (/), a subdirectory named **Oregon**, a subdirectory of the Oregon directory named **Portland**, and a text file in the Portland directory named **Data.txt**. Appearing in those columns are short form

representations of the ACL entry required to grant permissions. **N/A** (*Not applicable*) appears in the column if an ACL entry is not required to perform the operation.

Operation	**Assigned Azure role (with or without conditions)**	**/**	**Oregon/**	**Portland/**	**Data.txt**
Read Data.txt	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	N/A	N/A	N/A	N/A
	None	--X	--X	--X	R--
Append to Data.txt	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	--X	--X	--X	-W-
	None	--X	--X	--X	RW-
Delete Data.txt	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	--X	--X	-WX	N/A
	None	--X	--X	-WX	N/A
Create Data.txt	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	--X	--X	-WX	N/A
	None	--X	--X	-WX	N/A
List /	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	N/A	N/A	N/A	N/A

Operation	Assigned Azure role (with or without conditions)	/	Oregon/	Portland/	Data.txt
	None	R–X	N/A	N/A	N/A
List /Oregon/	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	N/A	N/A	N/A	N/A
	None	–––X	R–X	N/A	N/A
List /Oregon/Portland/	Storage Blob Data Owner	N/A	N/A	N/A	N/A
	Storage Blob Data Contributor	N/A	N/A	N/A	N/A
	Storage Blob Data Reader	N/A	N/A	N/A	N/A
	None	–––X	––X	R–X	N/A

> **Note**
>
> To view the contents of a container in Azure Storage Explorer, security principals must **sign in to Storage Explorer by using Azure AD**, and (at a minimum) have read access (R--) to the root folder (`\`) of a container. This level of permission does give them the ability to list the contents of the root folder. If you don't want the contents of the root folder to be visible, you can assign them **Reader** role. With that role, they'll be able to list the containers in the account, but not container contents. You can then grant access to specific directories and files by using ACLs.

Security groups

Always use Azure AD security groups as the assigned principal in an ACL entry. Resist the opportunity to directly assign individual users or service principals. Using this structure will allow you to add and remove users or service principals without the need to reapply ACLs to an entire directory structure. Instead, you can just add or remove users and service principals from the appropriate Azure AD security group.

There are many different ways to set up groups. For example, imagine that you have a directory named **/LogData** which holds log data that is generated by your server. Azure

Data Factory (ADF) ingests data into that folder. Specific users from the service engineering team will upload logs and manage other users of this folder, and various Databricks clusters will analyze logs from that folder.

To enable these activities, you could create a `LogsWriter` group and a `LogsReader` group. Then, you could assign permissions as follows:

- Add the `LogsWriter` group to the ACL of the **/LogData** directory with `rwx` permissions.
- Add the `LogsReader` group to the ACL of the **/LogData** directory with `r-x` permissions.
- Add the service principal object or Managed Service Identity (MSI) for ADF to the `LogsWriters` group.
- Add users in the service engineering team to the `LogsWriter` group.
- Add the service principal object or MSI for Databricks to the `LogsReader` group.

If a user in the service engineering team leaves the company, you could just remove them from the `LogsWriter` group. If you did not add that user to a group, but instead, you added a dedicated ACL entry for that user, you would have to remove that ACL entry from the **/LogData** directory. You would also have to remove the entry from all subdirectories and files in the entire directory hierarchy of the **/LogData** directory.

To create a group and add members, see Create a basic group and add members using Azure Active Directory.

> **ⓘ Important**
>
> Azure Data Lake Storage Gen2 depends on Azure Active Directory (Azure AD) to manage security groups. Azure AD recommends that you limit group membership for a given security principal to less than 200. This recommendation is due to a limitation of JSON Web Tokens (JWT) that provide a security principal's group membership information within Azure AD applications. Exceeding this limit might lead to unexpected performance issues with Data Lake Storage Gen2. To learn more, see **Configure group claims for applications by using Azure Active Directory**.

Limits on Azure role assignments and ACL entries

By using groups, you're less likely to exceed the maximum number of role assignments per subscription and the maximum number of ACL entries per file or directory. The following table describes these limits.

Mechanism	Scope	Limits	Supported level of permission
Azure RBAC	Storage accounts, containers. Cross resource Azure role assignments at subscription or resource group level.	4000 Azure role assignments in a subscription	Azure roles (built-in or custom)
ACL	Directory, file	32 ACL entries (effectively 28 ACL entries) per file and per directory. Access and default ACLs each have their own 32 ACL entry limit.	ACL permission

Shared Key and Shared Access Signature (SAS) authorization

Azure Data Lake Storage Gen2 also supports Shared Key and SAS methods for authentication. A characteristic of these authentication methods is that no identity is associated with the caller and therefore security principal permission-based authorization cannot be performed.

In the case of Shared Key, the caller effectively gains 'super-user' access, meaning full access to all operations on all resources including data, setting owner, and changing ACLs.

SAS tokens include allowed permissions as part of the token. The permissions included in the SAS token are effectively applied to all authorization decisions, but no additional ACL checks are performed.

Next steps

To learn more about access control lists, see Access control lists (ACLs) in Azure Data Lake Storage Gen2.

Azure services that support Azure Data Lake Storage Gen2

Article • 03/09/2023

You can use Azure services to ingest data, perform analytics, and create visual representations. This article provides a list of supported Azure services, discloses their level of support, and provides you with links to articles that help you to use these services with Azure Data Lake Storage Gen2.

Supported Azure services

This table lists the Azure services that you can use with Azure Data Lake Storage Gen2. The items that appear in these tables will change over time as support continues to expand.

> **Note**
>
> Support level refers only to how the service is supported with Data Lake Storage Gen 2.

Azure service	Support level	Azure AD	Shared Key	Related articles
Azure Data Factory	Generally available	Yes	Yes	• Load data into Azure Data Lake Storage Gen2 with Azure Data Factory
Azure Databricks	Generally available	Yes	Yes	• Use with Azure Databricks • Tutorial: Extract, transform, and load data by using Azure Databricks • Tutorial: Access Data Lake Storage Gen2 data with Azure Databricks using Spark
Azure Event Hubs	Generally available	No	Yes	• Capture events through Azure Event Hubs in Azure Blob Storage or Azure Data Lake Storage

Azure service	Support level	Azure AD	Shared Key	Related articles
Azure Event Grid	Generally available	Yes	Yes	• Tutorial: Implement the data lake capture pattern to update a Databricks Delta table
Azure Logic Apps	Generally available	No	Yes	• Overview - What is Azure Logic Apps?
Azure Machine Learning	Generally available	Yes	Yes	• Access data in Azure storage services
Azure Stream Analytics	Generally available	Yes	Yes	• Quickstart: Create a Stream Analytics job by using the Azure portal • Egress to Azure Data Lake Gen2
Data Box	Generally available	No	Yes	• Use Azure Data Box to migrate data from an on-premises HDFS store to Azure Storage
HDInsight	Generally available	Yes	Yes	• Azure Storage overview in HDInsight • Use Azure storage with Azure HDInsight clusters • Use Azure Data Lake Storage Gen2 with Azure HDInsight clusters • Using the HDFS CLI with Data Lake Storage Gen2 • Tutorial: Extract, transform, and load data by using Apache Hive on Azure HDInsight
IoT Hub	Generally available	Yes	Yes	• Use IoT Hub message routing to send device-to-cloud messages to different endpoints
Power BI	Generally available	Yes	Yes	• Analyze data in Data Lake Storage Gen2 using Power BI

Azure service	Support level	Azure AD	Shared Key	Related articles
Azure Synapse Analytics (formerly SQL Data Warehouse)	Generally available	Yes	Yes	• Analyze data in a storage account
SQL Server Integration Services (SSIS)	Generally available	Yes	Yes	• Azure Storage connection manager
Azure Data Explorer	Generally available	Yes	Yes	• Query data in Azure Data Lake using Azure Data Explorer
Azure Cognitive Search	Generally available	Yes	Yes	• Index and search Azure Data Lake Storage Gen2 documents
Azure SQL Managed Instance	Preview	No	Yes	• Data virtualization with Azure SQL Managed Instance

Tip

To see how services organized into categories such as ingest, download, process, and visualize, see **Ingest, process, and analyze**.

See also

- Known issues with Azure Data Lake Storage Gen2
- Blob Storage feature support in Azure Storage accounts
- Open source platforms that support Azure Data Lake Storage Gen2
- Multi-protocol access on Azure Data Lake Storage
- Best practices for using Azure Data Lake Storage Gen2