Contents

Give feedback on UserVoice

Microsoft Q&A question page

Pricing

Pricing calculator

Stack Overflow Forum

Videos

Introduction to Azure Data Lake Storage Gen2

10/3/2022 • 4 minutes to read • Edit Online

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'll also get low-cost, tiered storage, with high availability/disaster recovery capabilities.

Designed for enterprise big data analytics

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.

A fundamental part of Data Lake Storage Gen2 is the addition of a hierarchical namespace to Blob storage. The hierarchical namespace organizes objects/files into a hierarchy of directories for efficient data access. A common object store naming convention uses slashes in the name to mimic a hierarchical directory structure. This structure becomes real with Data Lake Storage Gen2. 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.

Data Lake Storage Gen2 builds on Blob storage and enhances performance, management, and security in the following ways:

- **Performance** is optimized because you don't need to copy or transform data as a prerequisite for analysis. Compared to the flat namespace on Blob storage, the hierarchical namespace greatly improves the performance of directory management operations, which improves overall job performance.
- **Management** is easier because you can organize and manipulate files through directories and subdirectories.
- **Security** is enforceable because you can define POSIX permissions on directories or individual files.

Also, Data Lake Storage Gen2 is very cost effective because it's built on top of the low-cost Azure Blob Storage. The extra features further lower the total cost of ownership for running big data analytics on Azure.

Key features of Data Lake Storage Gen2

- **Hadoop compatible access:** Data Lake Storage Gen2 allows you to manage and access data just as you would with a Hadoop Distributed File System (HDFS). The new ABFS driver (used to access data) is available within all Apache Hadoop environments. These environments include Azure HDInsight, Azure Databricks, and Azure Synapse Analytics.
- **A superset of POSIX permissions:** The security model for Data Lake Gen2 supports ACL and POSIX permissions along with some extra granularity specific to Data Lake Storage Gen2. Settings may be configured through Storage Explorer or through frameworks like Hive and Spark.
- **Cost-effective:** Data Lake Storage Gen2 offers low-cost storage capacity and transactions. Features such as Azure Blob Storage lifecycle optimize costs as data transitions through its lifecycle.
- **Optimized driver:** The ABFS driver is optimized specifically for big data analytics. The corresponding

REST APIs are surfaced through the endpoint `dfs.core.windows.net`.

Scalability

Azure Storage is scalable by design whether you access via Data Lake Storage Gen2 or Blob storage interfaces. It's able to store and serve *many exabytes of data*. This amount of storage is available with throughput measured in gigabits per second (Gbps) at high levels of input/output operations per second (IOPS). Processing is executed at near-constant per-request latencies that are measured at the service, account, and file levels.

Cost effectiveness

Because Data Lake Storage Gen2 is built on top of Azure Blob Storage, storage capacity and transaction costs are lower. Unlike other cloud storage services, you don't have to move or transform your data before you can analyze it. For more information about pricing, see Azure Storage pricing.

Additionally, features such as the hierarchical namespace significantly improve the overall performance of many analytics jobs. This improvement in performance means that you require less compute power to process the same amount of data, resulting in a lower total cost of ownership (TCO) for the end-to-end analytics job.

One service, multiple concepts

Because Data Lake Storage Gen2 is built on top of Azure Blob Storage, multiple concepts can describe the same, shared things.

The following are the equivalent entities, as described by different concepts. Unless specified otherwise these entities are directly synonymous:

CONCEPT	TOP LEVEL ORGANIZATION	LOWER LEVEL ORGANIZATION	DATA CONTAINER
Blobs - General purpose object storage	Container	Virtual directory (SDK only - doesn't provide atomic manipulation)	Blob
Azure Data Lake Storage Gen2 - Analytics Storage	Container	Directory	File

Supported Blob Storage features

Blob Storage features such as diagnostic logging, access tiers, and Blob Storage lifecycle management policies are available to your account. Most Blob Storage features are fully supported, but some features are supported only at the preview level or not yet supported.

To see how each Blob Storage feature is supported with Data Lake Storage Gen2, see Blob Storage feature support in Azure Storage accounts.

Supported Azure service integrations

Data Lake Storage gen2 supports several Azure services. You can use them 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.

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.

See also

- 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

What is Azure Data Lake Storage Gen1?

10/3/2022 • 5 minutes to read • Edit Online

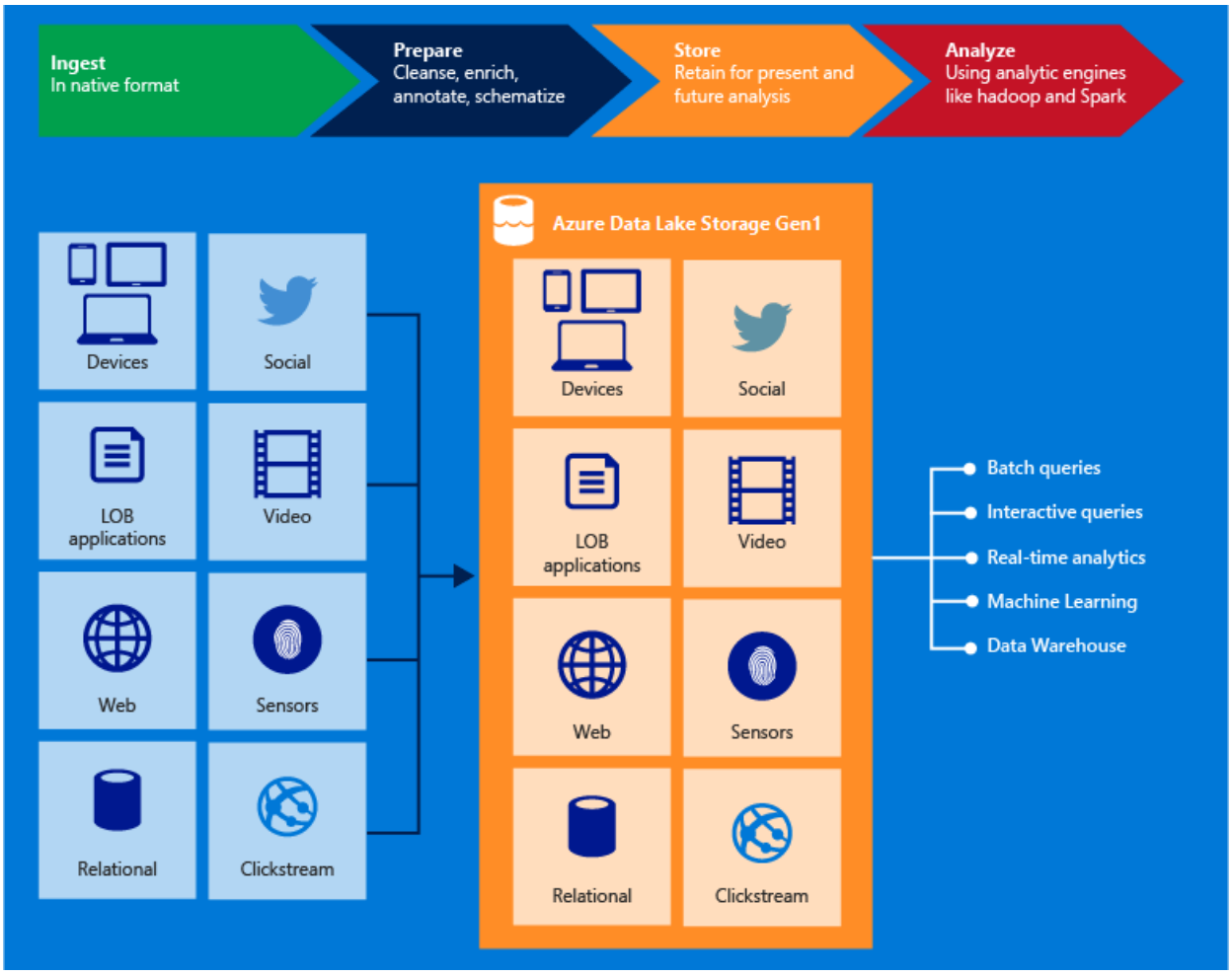
NOTE

On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.

Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

Azure Data Lake Storage Gen1 is an enterprise-wide hyper-scale repository for big data analytic workloads. Azure Data Lake enables you to capture data of any size, type, and ingestion speed in one single place for operational and exploratory analytics.

Data Lake Storage Gen1 can be accessed from Hadoop (available with HDInsight cluster) using the WebHDFS-compatible REST APIs. It's designed to enable analytics on the stored data and is tuned for performance for data analytics scenarios. Data Lake Storage Gen1 includes all enterprise-grade capabilities: security, manageability, scalability, reliability, and availability.



Key capabilities

Some of the key capabilities of Data Lake Storage Gen1 include the following.

Built for Hadoop

Data Lake Storage Gen1 is an Apache Hadoop file system that's compatible with Hadoop Distributed File System (HDFS), and works with the Hadoop ecosystem. Your existing HDInsight applications or services that use the WebHDFS API can easily integrate with Data Lake Storage Gen1. Data Lake Storage Gen1 also exposes a WebHDFS-compatible REST interface for applications.

You can easily analyze data stored in Data Lake Storage Gen1 using Hadoop analytic frameworks such as MapReduce or Hive. You can provision Azure HDInsight clusters and configure them to directly access data stored in Data Lake Storage Gen1.

Unlimited storage, petabyte files

Data Lake Storage Gen1 provides unlimited storage and can store a variety of data for analytics. It doesn't impose any limits on account sizes, file sizes, or the amount of data that can be stored in a data lake. Individual files can range from kilobyte to petabytes in size. Data is stored durably by making multiple copies. There is no limit on the duration of time for which the data can be stored in the data lake.

Performance-tuned for big data analytics

Data Lake Storage Gen1 is built for running large-scale analytic systems that require massive throughput to query and analyze large amounts of data. The data lake spreads parts of a file over a number of individual storage servers. This improves the read throughput when reading the file in parallel for performing data analytics.

Enterprise ready: Highly available and secure

Data Lake Storage Gen1 provides industry-standard availability and reliability. Your data assets are stored durably by making redundant copies to guard against any unexpected failures.

Data Lake Storage Gen1 also provides enterprise-grade security for the stored data. For more information, see Securing data in Azure Data Lake Storage Gen1.

All data

Data Lake Storage Gen1 can store any data in its native format, without requiring any prior transformations. Data Lake Storage Gen1 does not require a schema to be defined before the data is loaded, leaving it up to the individual analytic framework to interpret the data and define a schema at the time of the analysis. The ability to store files of arbitrary sizes and formats makes it possible for Data Lake Storage Gen1 to handle structured, semi-structured, and unstructured data.

Data Lake Storage Gen1 containers for data are essentially folders and files. You operate on the stored data using SDKs, the Azure portal, and Azure PowerShell. If you put your data into the store using these interfaces and using the appropriate containers, you can store any type of data. Data Lake Storage Gen1 does not perform any special handling of data based on the type of data it stores.

Securing data

Data Lake Storage Gen1 uses Azure Active Directory (Azure AD) for authentication, and access control lists (ACLs) to manage access to your data.

FEATURE	DESCRIPTION

FEATURE	DESCRIPTION
Authentication	Data Lake Storage Gen1 integrates with Azure AD for identity and access management for all the data stored in Data Lake Storage Gen1. Because of the integration, Data Lake Storage Gen1 benefits from all Azure AD feature such as multi-factor authentication, Conditional Access, Azure role-based access control, application usage monitoring, security monitoring and alerting, and so on. Data Lake Storage Gen1 supports the OAuth 2.0 protocol for authentication within the REST interface. See Data Lake Storage Gen1 authentication.
Access control	Data Lake Storage Gen1 provides access control by supporting POSIX-style permissions exposed by the WebHDFS protocol. You can enable ACLs on the root folder, on subfolders, and on individual files. For more information about how ACLs work in the context of Data Lake Storage Gen1, see Access control in Data Lake Storage Gen1.
Encryption	Data Lake Storage Gen1 also provides encryption for data that's stored in the account. You specify the encryption settings while creating a Data Lake Storage Gen1 account. You can choose to have your data encrypted or opt for no encryption. For more information, see Encryption in Data Lake Storage Gen1. For instructions on how to provide encryption-related configuration, see Get started with Data Lake Storage Gen1 using the Azure portal.

For instructions on how to secure data in Data Lake Storage Gen1, see Securing data in Azure Data Lake Storage Gen1.

Application compatibility

Data Lake Storage Gen1 is compatible with most open-source components in the Hadoop ecosystem. It also integrates well with other Azure services. To learn more about how you can use Data Lake Storage Gen1 with open-source components and other Azure services, use the following links:

- See Applications and services compatible with Azure Data Lake Storage Gen1 for a list of open-source applications interoperable with Data Lake Storage Gen1.
- See Integrating with other Azure services to understand how to use Data Lake Storage Gen1 with other Azure services to enable a wider range of scenarios.
- See Scenarios for using Data Lake Storage Gen1 to learn how to use Data Lake Storage Gen1 in scenarios such as ingesting data, processing data, downloading data, and visualizing data.

Data Lake Storage Gen1 file system

Data Lake Storage Gen1 can be accessed via the filesystem AzureDataLakeFilesystem (adl://) in Hadoop environments (available with HDInsight cluster). Applications and services that use adl:// can take advantage of further performance optimizations that aren't currently available in WebHDFS. As a result, Data Lake Storage Gen1 gives you the flexibility to either make use of the best performance with the recommended option of using adl:// or maintain existing code by continuing to use the WebHDFS API directly. Azure HDInsight fully leverages the AzureDataLakeFilesystem to provide the best performance on Data Lake Storage Gen1.

You can access your data in Data Lake Storage Gen1 using `adl://<data_lake_storage_gen1_name>.azuredatalakestore.net`. For more information about how to access the data in Data Lake Storage Gen1, see View properties of the stored data.

Next steps

- Get started with Data Lake Storage Gen1 using the Azure portal
- Get started with Data Lake Storage Gen1 using .NET SDK
- Use Azure HDInsight with Data Lake Storage Gen1

Comparing Azure Data Lake Storage Gen1 and Azure Blob Storage

10/3/2022 • 2 minutes to read • Edit Online

NOTE

On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.

Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

The table in this article summarizes the differences between Azure Data Lake Storage Gen1 and Azure Blob Storage along some key aspects of big data processing. Azure Blob Storage is a general purpose, scalable object store that is designed for a wide variety of storage scenarios. Azure Data Lake Storage Gen1 is a hyper-scale repository that is optimized for big data analytics workloads.

CATEGORY	AZURE DATA LAKE STORAGE GEN1	AZURE BLOB STORAGE
Purpose	Optimized storage for big data analytics workloads	General purpose object store for a wide variety of storage scenarios, including big data analytics
Use Cases	Batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets	Any type of text or binary data, such as application back end, backup data, media storage for streaming and general purpose data. Additionally, full support for analytics workloads; batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets
Key Concepts	Data Lake Storage Gen1 account contains folders, which in turn contains data stored as files	Storage account has containers, which in turn has data in the form of blobs
Structure	Hierarchical file system	Object store with flat namespace
API	REST API over HTTPS	REST API over HTTP/HTTPS
Server-side API	WebHDFS-compatible REST API	Azure Blob Storage REST API
Hadoop File System Client	Yes	Yes
Data Operations - Authentication	Based on Azure Active Directory Identities	Based on shared secrets - Account Access Keys and Shared Access Signature Keys.

CATEGORY	AZURE DATA LAKE STORAGE GEN1	AZURE BLOB STORAGE
Data Operations - Authentication Protocol	OpenID Connect. Calls must contain a valid JWT (JSON web token) issued by Azure Active Directory.	Hash-based Message Authentication Code (HMAC). Calls must contain a Base64-encoded SHA-256 hash over a part of the HTTP request.
Data Operations - Authorization	POSIX Access Control Lists (ACLs). ACLs based on Azure Active Directory Identities can be set at the file and folder level.	For account-level authorization – Use Account Access Keys For account, container, or blob authorization - Use Shared Access Signature Keys
Data Operations - Auditing	Available. See here for information.	Available
Encryption data at rest	• Transparent, Server side • With service-managed keys • With customer-managed keys in Azure KeyVault	• Transparent, Server side • With service-managed keys • With customer-managed keys in Azure KeyVault (preview) • Client-side encryption
Management operations (for example, Account Create)	Azure role-based access control (Azure RBAC) for account management	Azure role-based access control (Azure RBAC) for account management
Developer SDKs	.NET, Java, Python, Node.js	.NET, Java, Python, Node.js, C++, Ruby, PHP, Go, Android, iOS
Analytics Workload Performance	Optimized performance for parallel analytics workloads. High Throughput and IOPS.	Optimized performance for parallel analytics workloads.
Size limits	No limits on account sizes, file sizes, or number of files	For specific limits, see Scalability targets for standard storage accounts and Scalability and performance targets for Blob storage. Larger account limits available by contacting Azure Support
Geo-redundancy	Locally redundant (multiple copies of data in one Azure region)	Locally redundant (LRS), zone redundant (ZRS), globally redundant (GRS), read-access globally redundant (RA-GRS). See here for more information
Service state	Generally available	Generally available
Regional availability	See here	Available in all Azure regions
Price	See Pricing	See Pricing

Using Azure Data Lake Storage Gen1 for big data requirements

10/3/2022 • 6 minutes to read • Edit Online

NOTE

On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.

Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

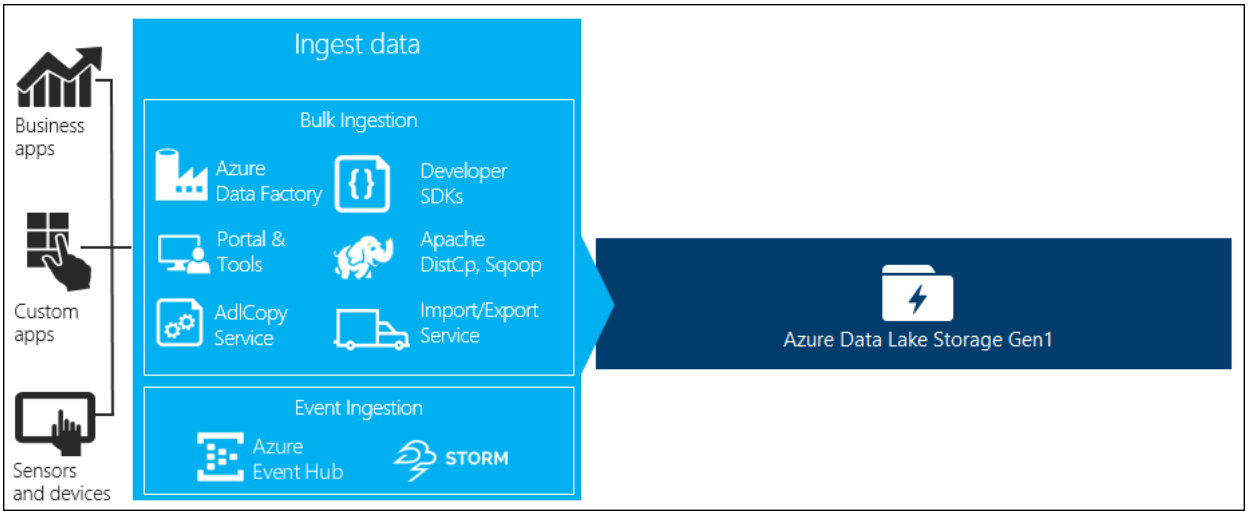
There are four key stages in big data processing:

- Ingesting large amounts of data into a data store, at real-time or in batches
- Processing the data
- Downloading the data
- Visualizing the data

In this article, we look at these stages with respect to Azure Data Lake Storage Gen1 to understand the options and tools available to meet your big data needs.

Ingest data into Data Lake Storage Gen1

This section highlights the different sources of data and the different ways in which that data can be ingested into a Data Lake Storage Gen1 account.



Ad hoc data

This represents smaller data sets that are used for prototyping a big data application. There are different ways of ingesting ad hoc data depending on the source of the data.

DATA SOURCE	INGEST IT USING

DATA SOURCE	INGEST IT USING
Local computer	• Azure portal • Azure PowerShell • Azure CLI • Using Data Lake Tools for Visual Studio
Azure Storage Blob	• Azure Data Factory • AdlCopy tool • DistCp running on HDInsight cluster

Streamed data

This represents data that can be generated by various sources such as applications, devices, sensors, etc. This data can be ingested into Data Lake Storage Gen1 by a variety of tools. These tools will usually capture and process the data on an event-by-event basis in real-time, and then write the events in batches into Data Lake Storage Gen1 so that they can be further processed.

Following are tools that you can use:

- Azure Stream Analytics - Events ingested into Event Hubs can be written to Azure Data Lake Storage Gen1 using an Azure Data Lake Storage Gen1 output.
- EventProcessorHost – You can receive events from Event Hubs and then write it to Data Lake Storage Gen1 using the Data Lake Storage Gen1 .NET SDK.

Relational data

You can also source data from relational databases. Over a period of time, relational databases collect huge amounts of data which can provide key insights if processed through a big data pipeline. You can use the following tools to move such data into Data Lake Storage Gen1.

- Apache Sqoop
- Azure Data Factory

Web server log data (upload using custom applications)

This type of dataset is specifically called out because analysis of web server log data is a common use case for big data applications and requires large volumes of log files to be uploaded to Data Lake Storage Gen1. You can use any of the following tools to write your own scripts or applications to upload such data.

- Azure CLI
- Azure PowerShell
- Azure Data Lake Storage Gen1 .NET SDK
- Azure Data Factory

For uploading web server log data, and also for uploading other kinds of data (e.g. social sentiments data), it is a good approach to write your own custom scripts/applications because it gives you the flexibility to include your data uploading component as part of your larger big data application. In some cases this code may take the form of a script or simple command line utility. In other cases, the code may be used to integrate big data processing into a business application or solution.

Data associated with Azure HDInsight clusters

Most HDInsight cluster types (Hadoop, HBase, Storm) support Data Lake Storage Gen1 as a data storage repository. HDInsight clusters access data from Azure Storage Blobs (WASB). For better performance, you can copy the data from WASB into a Data Lake Storage Gen1 account associated with the cluster. You can use the following tools to copy the data.

- Apache DistCp
- AdlCopy Service
- Azure Data Factory

Data stored in on-premises or IaaS Hadoop clusters

Large amounts of data may be stored in existing Hadoop clusters, locally on machines using HDFS. The Hadoop clusters may be in an on-premises deployment or may be within an IaaS cluster on Azure. There could be requirements to copy such data to Azure Data Lake Storage Gen1 for a one-off approach or in a recurring fashion. There are various options that you can use to achieve this. Below is a list of alternatives and the associated trade-offs.

APPROACH	DETAILS	ADVANTAGES	CONSIDERATIONS
Use Azure Data Factory (ADF) to copy data directly from Hadoop clusters to Azure Data Lake Storage Gen1	ADF supports HDFS as a data source	ADF provides out-of-the-box support for HDFS and first class end-to-end management and monitoring	Requires Data Management Gateway to be deployed on-premises or in the IaaS cluster
Export data from Hadoop as files. Then copy the files to Azure Data Lake Storage Gen1 using appropriate mechanism.	You can copy files to Azure Data Lake Storage Gen1 using: • Azure PowerShell for Windows OS • Azure CLI • Custom app using any Data Lake Storage Gen1 SDK	Quick to get started. Can do customized uploads	Multi-step process that involves multiple technologies. Management and monitoring will grow to be a challenge over time given the customized nature of the tools
Use Distcp to copy data from Hadoop to Azure Storage. Then copy data from Azure Storage to Data Lake Storage Gen1 using appropriate mechanism.	You can copy data from Azure Storage to Data Lake Storage Gen1 using: • Azure Data Factory • AdlCopy tool • Apache DistCp running on HDInsight clusters	You can use open-source tools.	Multi-step process that involves multiple technologies

Really large datasets

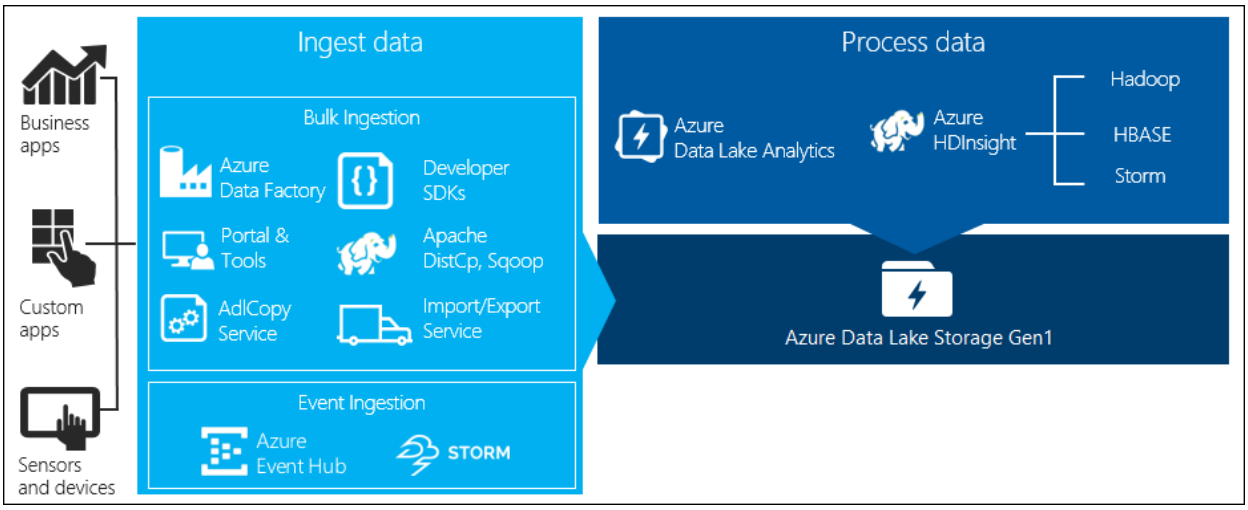
For uploading datasets that range in several terabytes, using the methods described above can sometimes be slow and costly. In such cases, you can use the options below.

- **Using Azure ExpressRoute**. Azure ExpressRoute lets you create private connections between Azure datacenters and infrastructure on your premises. This provides a reliable option for transferring large amounts of data. For more information, see Azure ExpressRoute documentation.
- **"Offline" upload of data**. If using Azure ExpressRoute is not feasible for any reason, you can use Azure Import/Export service to ship hard disk drives with your data to an Azure data center. Your data is first uploaded to Azure Storage Blobs. You can then use Azure Data Factory or AdlCopy tool to copy data from Azure Storage Blobs to Data Lake Storage Gen1.

  > **NOTE**
  > While using the Import/Export service, the file sizes on the disks that you ship to Azure data center should not be greater than 195 GB.

Process data stored in Data Lake Storage Gen1

Once the data is available in Data Lake Storage Gen1 you can run analysis on that data using the supported big data applications. Currently, you can use Azure HDInsight and Azure Data Lake Analytics to run data analysis jobs on the data stored in Data Lake Storage Gen1.



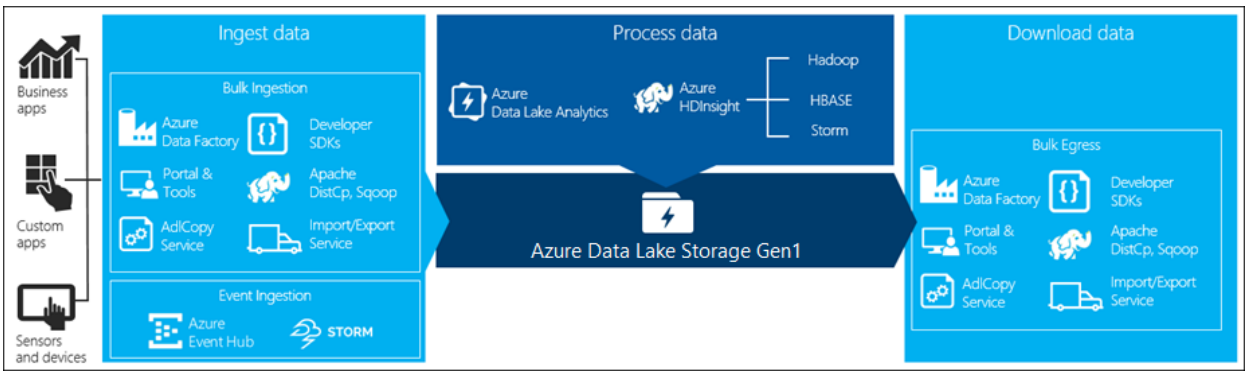
You can look at the following examples.

- Create an HDInsight cluster with Data Lake Storage Gen1 as storage
- Use Azure Data Lake Analytics with Data Lake Storage Gen1

Download data from Data Lake Storage Gen1

You might also want to download or move data from Azure Data Lake Storage Gen1 for scenarios such as:

- Move data to other repositories to interface with your existing data processing pipelines. For example, you might want to move data from Data Lake Storage Gen1 to Azure SQL Database or SQL Server.
- Download data to your local computer for processing in IDE environments while building application prototypes.



In such cases, you can use any of the following options:

- Apache Sqoop
- Azure Data Factory
- Apache DistCp

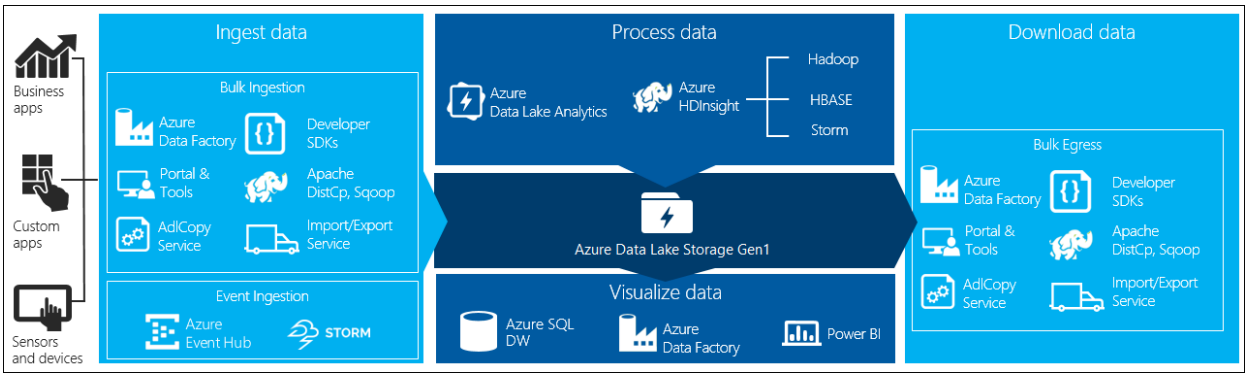
You can also use the following methods to write your own script/application to download data from Data Lake Storage Gen1.

- Azure CLI
- Azure PowerShell

- Azure Data Lake Storage Gen1 .NET SDK

Visualize data in Data Lake Storage Gen1

You can use a mix of services to create visual representations of data stored in Data Lake Storage Gen1.



- You can start by using Azure Data Factory to move data from Data Lake Storage Gen1 to Azure Synapse Analytics
- After that, you can integrate Power BI with Azure Synapse Analytics to create visual representation of the data.

Open Source Big Data applications that work with Azure Data Lake Storage Gen1

10/3/2022 • 2 minutes to read • Edit Online

> **NOTE**
>
> On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.
>
> Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

This article lists the open source big data applications that work with Azure Data Lake Storage Gen1. For the applications in the table below, only the versions available with the listed distribution are supported. For information on what versions of these applications are available with HDInsight, see HDInsight component versioning.

OPEN SOURCE SOFTWARE	DISTRIBUTION
Apache Sqoop	HDInsight 3.2, 3.4, 3.5, and 3.6
MapReduce	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Storm	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Hive	HDInsight 3.2, 3.4, 3.5, and 3.6
HCatalog	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Mahout	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Pig/Pig Latin	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Oozie	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Zookeeper	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Tez	HDInsight 3.2, 3.4, 3.5, and 3.6
Apache Spark	HDInsight 3.4, 3.5, and 3.6

See also

- Overview of Azure Data Lake Storage Gen1

Best practices for using Azure Data Lake Storage Gen1

10/3/2022 • 19 minutes to read • Edit Online

NOTE

On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.

Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

In this article, you learn about best practices and considerations for working with Azure Data Lake Storage Gen1. This article provides information around security, performance, resiliency, and monitoring for Data Lake Storage Gen1. Before Data Lake Storage Gen1, working with truly big data in services like Azure HDInsight was complex. You had to shard data across multiple Blob storage accounts so that petabyte storage and optimal performance at that scale could be achieved. With Data Lake Storage Gen1, most of the hard limits for size and performance are removed. However, there are still some considerations that this article covers so that you can get the best performance with Data Lake Storage Gen1.

Security considerations

Azure Data Lake Storage Gen1 offers POSIX access controls and detailed auditing for Azure Active Directory (Azure AD) users, groups, and service principals. These access controls can be set to existing files and folders. The access controls can also be used to create defaults that can be applied to new files or folders. When permissions are set to existing folders and child objects, the permissions need to be propagated recursively on each object. If there are large number of files, propagating the permissions can take a long time. The time taken can range between 30-50 objects processed per second. Hence, plan the folder structure and user groups appropriately. Otherwise, it can cause unanticipated delays and issues when you work with your data.

Assume you have a folder with 100,000 child objects. If you take the lower bound of 30 objects processed per second, to update the permission for the whole folder could take an hour. More details on Data Lake Storage Gen1 ACLs are available at Access control in Azure Data Lake Storage Gen1. For improved performance on assigning ACLs recursively, you can use the Azure Data Lake Command-Line Tool. The tool creates multiple threads and recursive navigation logic to quickly apply ACLs to millions of files. The tool is available for Linux and Windows, and the documentation and downloads for this tool can be found on GitHub. These same performance improvements can be enabled by your own tools written with the Data Lake Storage Gen1 .NET and Java SDKs.

Use security groups versus individual users

When working with big data in Data Lake Storage Gen1, most likely a service principal is used to allow services such as Azure HDInsight to work with the data. However, there might be cases where individual users need access to the data as well. In such cases, you must use Azure Active Directory security groups instead of assigning individual users to folders and files.

Once a security group is assigned permissions, adding or removing users from the group doesn't require any updates to Data Lake Storage Gen1. This also helps ensure you don't exceed the limit of 32 Access and Default ACLs (this includes the four POSIX-style ACLs that are always associated with every file and folder: the owning user, the owning group, the mask, and other).

Security for groups

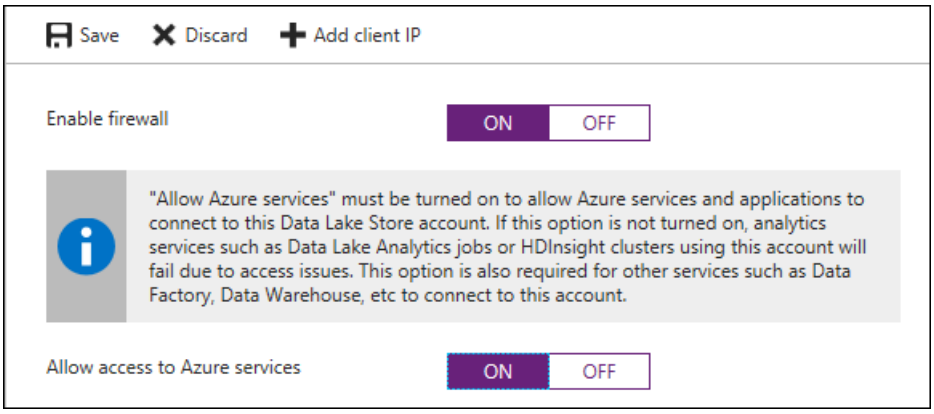
As discussed, when users need access to Data Lake Storage Gen1, it's best to use Azure Active Directory security groups. Some recommended groups to start with might be **ReadOnlyUsers**, **WriteAccessUsers**, and **FullAccessUsers** for the root of the account, and even separate ones for key subfolders. If there are any other anticipated groups of users that might be added later, but have not been identified yet, you might consider creating dummy security groups that have access to certain folders. Using security group ensures that later you do not need a long processing time for assigning new permissions to thousands of files.

Security for service principals

Azure Active Directory service principals are typically used by services like Azure HDInsight to access data in Data Lake Storage Gen1. Depending on the access requirements across multiple workloads, there might be some considerations to ensure security inside and outside of the organization. For many customers, a single Azure Active Directory service principal might be adequate, and it can have full permissions at the root of the Data Lake Storage Gen1 account. Other customers might require multiple clusters with different service principals where one cluster has full access to the data, and another cluster with only read access. As with the security groups, you might consider making a service principal for each anticipated scenario (read, write, full) once a Data Lake Storage Gen1 account is created.

Enable the Data Lake Storage Gen1 firewall with Azure service access

Data Lake Storage Gen1 supports the option of turning on a firewall and limiting access only to Azure services, which is recommended for a smaller attack vector from outside intrusions. Firewall can be enabled on the Data Lake Storage Gen1 account in the Azure portal via the **Firewall** > **Enable Firewall (ON)** > **Allow access to Azure services** options.



Once firewall is enabled, only Azure services such as HDInsight, Data Factory, Azure Synapse Analytics, etc. have access to Data Lake Storage Gen1. Due to the internal network address translation used by Azure, the Data Lake Storage Gen1 firewall does not support restricting specific services by IP and is only intended for restrictions of endpoints outside of Azure, such as on-premises.

Performance and scale considerations

One of the most powerful features of Data Lake Storage Gen1 is that it removes the hard limits on data throughput. Removing the limits enables customers to grow their data size and accompanied performance requirements without needing to shard the data. One of the most important considerations for optimizing Data Lake Storage Gen1 performance is that it performs the best when given parallelism.

Improve throughput with parallelism

Consider giving 8-12 threads per core for the most optimal read/write throughput. This is due to blocking reads/writes on a single thread, and more threads can allow higher concurrency on the VM. To ensure that levels are healthy and parallelism can be increased, be sure to monitor the VM's CPU utilization.

Avoid small file sizes

POSIX permissions and auditing in Data Lake Storage Gen1 comes with an overhead that becomes apparent

when working with numerous small files. As a best practice, you must batch your data into larger files versus writing thousands or millions of small files to Data Lake Storage Gen1. Avoiding small file sizes can have multiple benefits, such as:

- Lowering the authentication checks across multiple files
- Reduced open file connections
- Faster copying/replication
- Fewer files to process when updating Data Lake Storage Gen1 POSIX permissions

Depending on what services and workloads are using the data, a good size to consider for files is 256 MB or greater. If the file sizes cannot be batched when landing in Data Lake Storage Gen1, you can have a separate compaction job that combines these files into larger ones. For more information and recommendation on file sizes and organizing the data in Data Lake Storage Gen1, see Structure your data set.

Large file sizes and potential performance impact

Although Data Lake Storage Gen1 supports large files up to petabytes in size, for optimal performance and depending on the process reading the data, it might not be ideal to go above 2 GB on average. For example, when using **Distcp** to copy data between locations or different storage accounts, files are the finest level of granularity used to determine map tasks. So, if you are copying 10 files that are 1 TB each, at most 10 mappers are allocated. Also, if you have lots of files with mappers assigned, initially the mappers work in parallel to move large files. However, as the job starts to wind down only a few mappers remain allocated and you can be stuck with a single mapper assigned to a large file. Microsoft has submitted improvements to Distcp to address this issue in future Hadoop versions.

Another example to consider is when using Azure Data Lake Analytics with Data Lake Storage Gen1. Depending on the processing done by the extractor, some files that cannot be split (for example, XML, JSON) could suffer in performance when greater than 2 GB. In cases where files can be split by an extractor (for example, CSV), large files are preferred.

Capacity plan for your workload

Azure Data Lake Storage Gen1 removes the hard IO throttling limits that are placed on Blob storage accounts. However, there are still soft limits that need to be considered. The default ingress/egress throttling limits meet the needs of most scenarios. If your workload needs to have the limits increased, work with Microsoft support. Also, look at the limits during the proof-of-concept stage so that IO throttling limits are not hit during production. If that happens, it might require waiting for a manual increase from the Microsoft engineering team. If IO throttling occurs, Azure Data Lake Storage Gen1 returns an error code of 429, and ideally should be retried with an appropriate exponential backoff policy.

Optimize "writes" with the Data Lake Storage Gen1 driver buffer

To optimize performance and reduce IOPS when writing to Data Lake Storage Gen1 from Hadoop, perform write operations as close to the Data Lake Storage Gen1 driver buffer size as possible. Try not to exceed the buffer size before flushing, such as when streaming using Apache Storm or Spark streaming workloads. When writing to Data Lake Storage Gen1 from HDInsight/Hadoop, it is important to know that Data Lake Storage Gen1 has a driver with a 4-MB buffer. Like many file system drivers, this buffer can be manually flushed before reaching the 4-MB size. If not, it is immediately flushed to storage if the next write exceeds the buffer's maximum size. Where possible, you must avoid an overrun or a significant underrun of the buffer when syncing/flushing policy by count or time window.

Resiliency considerations

When architecting a system with Data Lake Storage Gen1 or any cloud service, you must consider your availability requirements and how to respond to potential interruptions in the service. An issue could be localized to the specific instance or even region-wide, so having a plan for both is important. Depending on the **recovery time objective** and the **recovery point objective** SLAs for your workload, you might choose a

more or less aggressive strategy for high availability and disaster recovery.

High availability and disaster recovery

High availability (HA) and disaster recovery (DR) can sometimes be combined together, although each has a slightly different strategy, especially when it comes to data. Data Lake Storage Gen1 already handles 3x replication under the hood to guard against localized hardware failures. However, since replication across regions is not built in, you must manage this yourself. When building a plan for HA, in the event of a service interruption the workload needs access to the latest data as quickly as possible by switching over to a separately replicated instance locally or in a new region.

In a DR strategy, to prepare for the unlikely event of a catastrophic failure of a region, it is also important to have data replicated to a different region. This data might initially be the same as the replicated HA data. However, you must also consider your requirements for edge cases such as data corruption where you may want to create periodic snapshots to fall back to. Depending on the importance and size of the data, consider rolling delta snapshots of 1-, 6-, and 24-hour periods on the local and/or secondary store, according to risk tolerances.

For data resiliency with Data Lake Storage Gen1, it is recommended to geo-replicate your data to a separate region with a frequency that satisfies your HA/DR requirements, ideally every hour. This frequency of replication minimizes massive data movements that can have competing throughput needs with the main system and a better recovery point objective (RPO). Additionally, you should consider ways for the application using Data Lake Storage Gen1 to automatically fail over to the secondary account through monitoring triggers or length of failed attempts, or at least send a notification to admins for manual intervention. Keep in mind that there is tradeoff of failing over versus waiting for a service to come back online. If the data hasn't finished replicating, a failover could cause potential data loss, inconsistency, or complex merging of the data.

Below are the top three recommended options for orchestrating replication between Data Lake Storage Gen1 accounts, and key differences between each of them.

	DISTCP	AZURE DATA FACTORY	ADLCOPY
Scale limits	Bounded by worker nodes	Limited by Max Cloud Data Movement units	Bound by Analytics units
Supports copying deltas	Yes	No	No
Built-in orchestration	No (use Oozie Airflow or cron jobs)	Yes	No (Use Azure Automation or Windows Task Scheduler)
Supported file systems	ADL, HDFS, WASB, S3, GS, CFS	Numerous, see Connectors.	ADL to ADL, WASB to ADL (same region only)
OS support	Any OS running Hadoop	N/A	Windows 10

Use Distcp for data movement between two locations

Short for distributed copy, Distcp is a Linux command-line tool that comes with Hadoop and provides distributed data movement between two locations. The two locations can be Data Lake Storage Gen1, HDFS, WASB, or S3. This tool uses MapReduce jobs on a Hadoop cluster (for example, HDInsight) to scale out on all the nodes. Distcp is considered the fastest way to move big data without special network compression appliances. Distcp also provides an option to only update deltas between two locations, handles automatic retries, as well as dynamic scaling of compute. This approach is incredibly efficient when it comes to replicating things like Hive/Spark tables that can have many large files in a single directory and you only want to copy over the modified data. For these reasons, Distcp is the most recommended tool for copying data between big data stores.

Copy jobs can be triggered by Apache Oozie workflows using frequency or data triggers, as well as Linux cron

jobs. For intensive replication jobs, it is recommended to spin up a separate HDInsight Hadoop cluster that can be tuned and scaled specifically for the copy jobs. This ensures that copy jobs do not interfere with critical jobs. If running replication on a wide enough frequency, the cluster can even be taken down between each job. If failing over to secondary region, make sure that another cluster is also spun up in the secondary region to replicate new data back to the primary Data Lake Storage Gen1 account once it comes back up. For examples of using Distcp, see Use Distcp to copy data between Azure Storage Blobs and Data Lake Storage Gen1.

Use Azure Data Factory to schedule copy jobs

Azure Data Factory can also be used to schedule copy jobs using a **Copy Activity**, and can even be set up on a frequency via the **Copy Wizard**. Keep in mind that Azure Data Factory has a limit of cloud data movement units (DMUs), and eventually caps the throughput/compute for large data workloads. Additionally, Azure Data Factory currently does not offer delta updates between Data Lake Storage Gen1 accounts, so folders like Hive tables would require a complete copy to replicate. Refer to the Copy Activity tuning guide for more information on copying with Data Factory.

AdlCopy

AdlCopy is a Windows command-line tool that allows you to copy data between two Data Lake Storage Gen1 accounts only within the same region. The AdlCopy tool provides a standalone option or the option to use an Azure Data Lake Analytics account to run your copy job. Though it was originally built for on-demand copies as opposed to a robust replication, it provides another option to do distributed copying across Data Lake Storage Gen1 accounts within the same region. For reliability, it's recommended to use the premium Data Lake Analytics option for any production workload. The standalone version can return busy responses and has limited scale and monitoring.

Like Distcp, the AdlCopy needs to be orchestrated by something like Azure Automation or Windows Task Scheduler. As with Data Factory, AdlCopy does not support copying only updated files, but recopies and overwrite existing files. For more information and examples of using AdlCopy, see Copy data from Azure Storage Blobs to Data Lake Storage Gen1.

Monitoring considerations

Data Lake Storage Gen1 provides detailed diagnostic logs and auditing. Data Lake Storage Gen1 provides some basic metrics in the Azure portal under the Data Lake Storage Gen1 account and in Azure Monitor. Availability of Data Lake Storage Gen1 is displayed in the Azure portal. However, this metric is refreshed every seven minutes and cannot be queried through a publicly exposed API. To get the most up-to-date availability of a Data Lake Storage Gen1 account, you must run your own synthetic tests to validate availability. Other metrics such as total storage utilization, read/write requests, and ingress/egress can take up to 24 hours to refresh. So, more up-to-date metrics must be calculated manually through Hadoop command-line tools or aggregating log information. The quickest way to get the most recent storage utilization is running this HDFS command from a Hadoop cluster node (for example, head node):

```
hdfs dfs -du -s -h adl://<adlsg1_account_name>.azuredatalakestore.net:443/
```

Export Data Lake Storage Gen1 diagnostics

One of the quickest ways to get access to searchable logs from Data Lake Storage Gen1 is to enable log shipping to **Log Analytics** under the **Diagnostics** blade for the Data Lake Storage Gen1 account. This provides immediate access to incoming logs with time and content filters, along with alerting options (email/webhook) triggered within 15-minute intervals. For instructions, see Accessing diagnostic logs for Azure Data Lake Storage Gen1.

For more real-time alerting and more control on where to land the logs, consider exporting logs to Azure EventHub where content can be analyzed individually or over a time window in order to submit real-time notifications to a queue. A separate application such as a Logic App can then consume and communicate the

alerts to the appropriate channel, as well as submit metrics to monitoring tools like NewRelic, Datadog, or AppDynamics. Alternatively, if you are using a third-party tool such as ElasticSearch, you can export the logs to Blob Storage and use the Azure Logstash plugin to consume the data into your Elasticsearch, Kibana, and Logstash (ELK) stack.

Turn on debug-level logging in HDInsight

If Data Lake Storage Gen1 log shipping is not turned on, Azure HDInsight also provides a way to turn on client-side logging for Data Lake Storage Gen1 via log4j. You must set the following property in **Ambari** > **YARN** > **Config** > **Advanced yarn-log4j configurations**:

```
log4j.logger.com.microsoft.azure.datalake.store=DEBUG
```

Once the property is set and the nodes are restarted, Data Lake Storage Gen1 diagnostics is written to the YARN logs on the nodes (/tmp/<user>/yarn.log), and important details like errors or throttling (HTTP 429 error code) can be monitored. This same information can also be monitored in Azure Monitor logs or wherever logs are shipped to in the Diagnostics blade of the Data Lake Storage Gen1 account. It is recommended to at least have client-side logging turned on or utilize the log shipping option with Data Lake Storage Gen1 for operational visibility and easier debugging.

Run synthetic transactions

Currently, the service availability metric for Data Lake Storage Gen1 in the Azure portal has 7-minute refresh window. Also, it cannot be queried using a publicly exposed API. Hence, it is recommended to build a basic application that does synthetic transactions to Data Lake Storage Gen1 that can provide up to the minute availability. An example might be creating a WebJob, Logic App, or Azure Function App to perform a read, create, and update against Data Lake Storage Gen1 and send the results to your monitoring solution. The operations can be done in a temporary folder and then deleted after the test, which might be run every 30-60 seconds, depending on requirements.

Directory layout considerations

When landing data into a data lake, it's important to pre-plan the structure of the data so that security, partitioning, and processing can be utilized effectively. Many of the following recommendations can be used whether it's with Azure Data Lake Storage Gen1, Blob Storage, or HDFS. Every workload has different requirements on how the data is consumed, but below are some common layouts to consider when working with IoT and batch scenarios.

IoT structure

In IoT workloads, there can be a great deal of data being landed in the data store 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/
```

There's an important reason to put the date at the end of the folder structure. If you want to lock down certain regions or subject matters to users/groups, then you can easily do so with the POSIX permissions. Otherwise, if there was a need to restrict a certain security group to viewing just the UK data or certain planes, with the date structure in front a separate permission would be required for numerous folders under every hour folder. Additionally, having the date structure in front would exponentially increase the number of folders as time went

on.

Batch jobs structure

From a high-level, a commonly used approach in batch processing is to land data in an "in" folder. Then, once the data is processed, put the new data into an "out" folder for downstream processes to consume. This directory structure is seen sometimes 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 folders for things such as region and subject matters (for example, organization, product/producer). This structure helps with securing the data across your organization and better management of the data in your workloads. Furthermore, 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, 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** folder since 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 folders, and orchestration by something like 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.

Next steps

- Overview of Azure Data Lake Storage Gen1
- Access Control in Azure Data Lake Storage Gen1
- Security in Azure Data Lake Storage Gen1
- Tuning Azure Data Lake Storage Gen1 for performance
- Performance tuning guidance for using HDInsight Spark with Azure Data Lake Storage Gen1
- Performance tuning guidance for using HDInsight Hive with Azure Data Lake Storage Gen1
- Create HDInsight clusters with Data Lake Storage Gen1

Get started with Azure Data Lake Storage Gen1 using the Azure portal

10/3/2022 • 5 minutes to read • Edit Online

> **NOTE**
>
> On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.
>
> Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

Learn how to use the Azure portal to create a Data Lake Storage Gen1 account and perform basic operations such as create folders, upload, and download data files, delete your account, etc. For more information, see Overview of Azure Data Lake Storage Gen1.

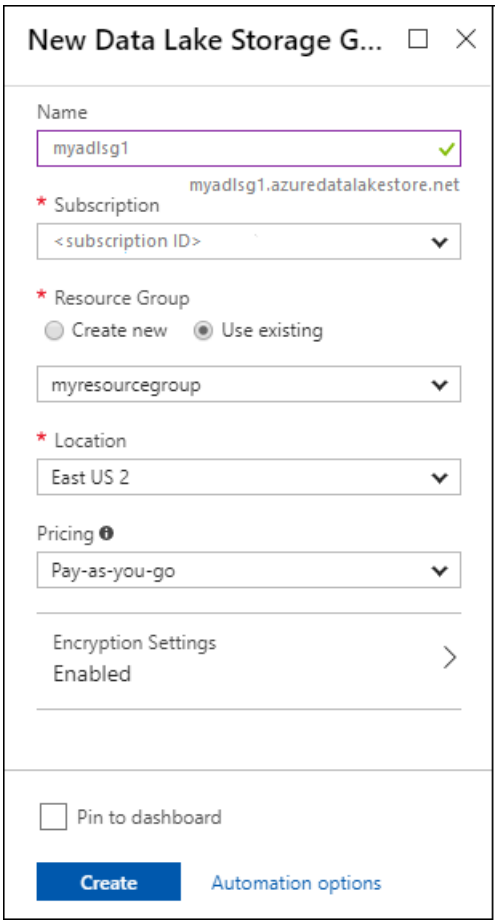
Prerequisites

Before you begin this tutorial, you must have the following items:

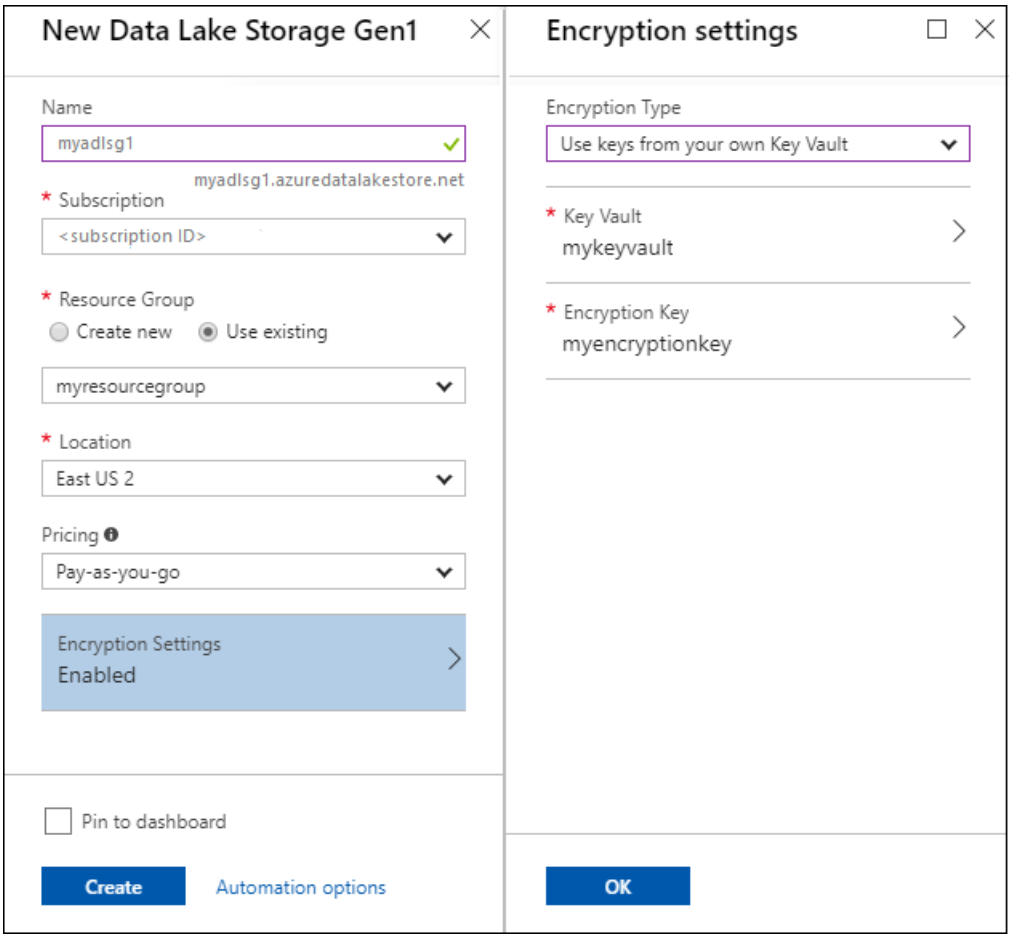
- **An Azure subscription**. See Get Azure free trial.

Create a Data Lake Storage Gen1 account

1. Sign on to the new Azure portal.
2. Click **Create a resource** > **Storage** > **Data Lake Storage Gen1**.
3. In the **New Data Lake Storage Gen1** blade, provide the values as shown in the following screenshot:



- **Name**. Enter a unique name for the Data Lake Storage Gen1 account.
- **Subscription**. Select the subscription under which you want to create a new Data Lake Storage Gen1 account.
- **Resource Group**. Select an existing resource group, or select the **Create new** option to create one. A resource group is a container that holds related resources for an application. For more information, see Resource Groups in Azure.
- **Location**: Select a location where you want to create the Data Lake Storage Gen1 account.
- **Encryption Settings**. There are three options:
  - **Do not enable encryption**.
  - **Use keys managed by Data Lake Storage Gen1**, if you want Data Lake Storage Gen1 to manage your encryption keys.
  - **Use keys from your own Key Vault**. You can select an existing Azure Key Vault or create a new Key Vault. To use the keys from a Key Vault, you must assign permissions for the Data Lake Storage Gen1 account to access the Azure Key Vault. For the instructions, see Assign permissions to Azure Key Vault.



Click **OK** in the **Encryption Settings** blade.

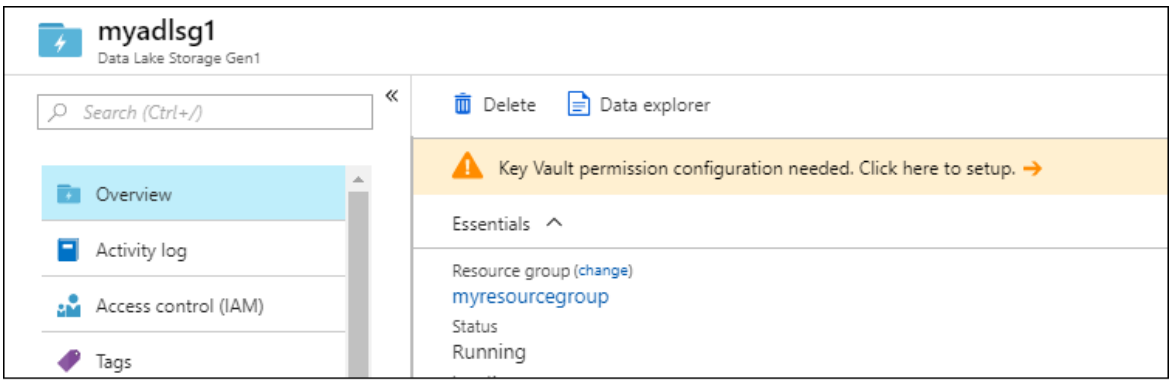
For more information, see Encryption of data in Azure Data Lake Storage Gen1.

4. Click **Create**. If you chose to pin the account to the dashboard, you are taken back to the dashboard and you can see the progress of your Data Lake Storage Gen1 account provisioning. Once the Data Lake Storage Gen1 account is provisioned, the account blade shows up.

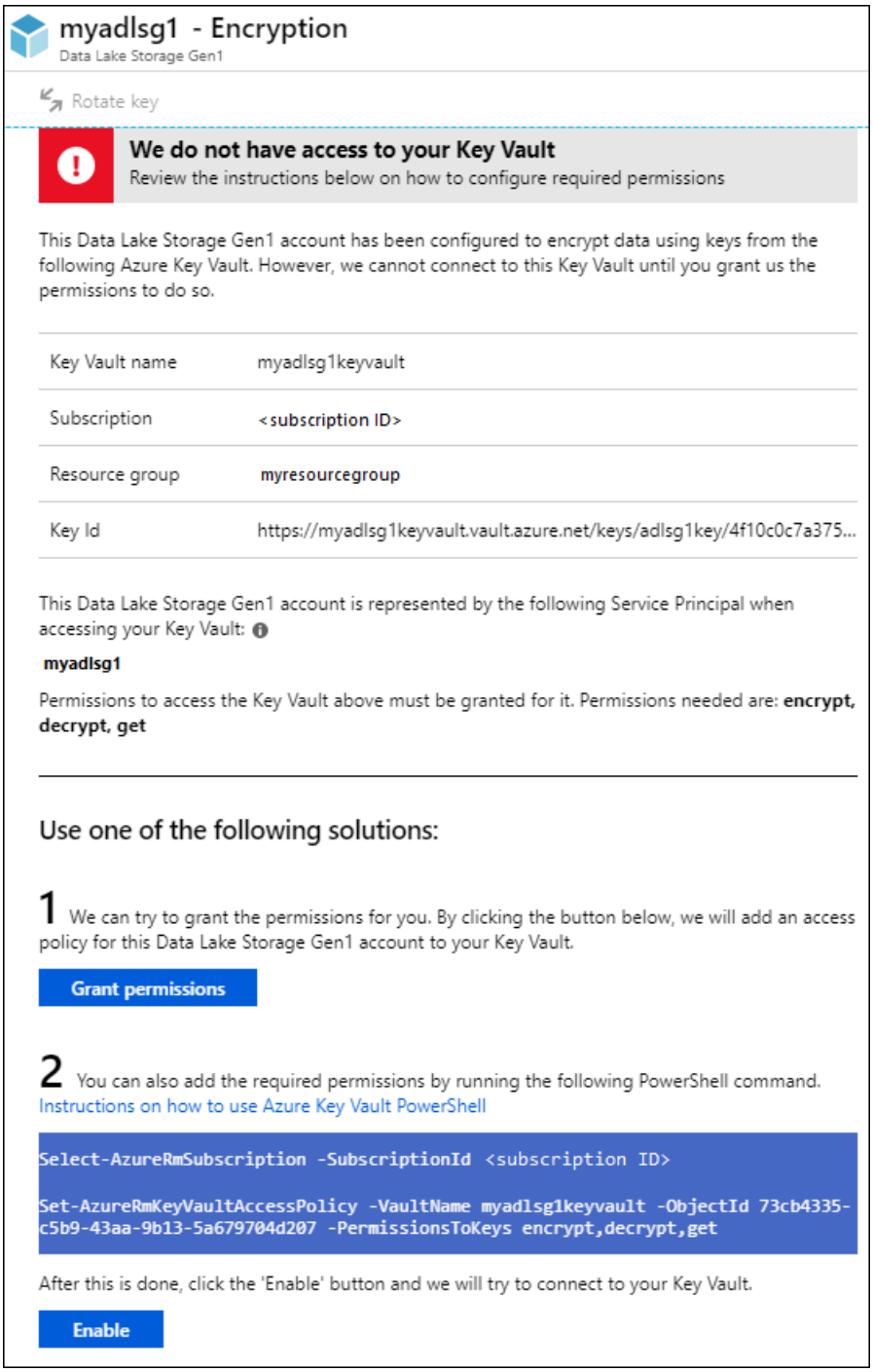
Assign permissions to Azure Key Vault

If you used keys from an Azure Key Vault to configure encryption on the Data Lake Storage Gen1 account, you must configure access between the Data Lake Storage Gen1 account and the Azure Key Vault account. Perform the following steps to do so.

1. If you used keys from the Azure Key Vault, the blade for the Data Lake Storage Gen1 account displays a warning at the top. Click the warning to open **Encryption**.



2. The blade shows two options to configure access.



- In the first option, click **Grant Permissions** to configure access. The first option is enabled only when the user that created the Data Lake Storage Gen1 account is also an admin for the Azure Key Vault.
- The other option is to run the PowerShell cmdlet displayed on the blade. You need to be the owner of the Azure Key Vault or have the ability to grant permissions on the Azure Key Vault. After you have run the cmdlet, come back to the blade and click **Enable** to configure access.

NOTE

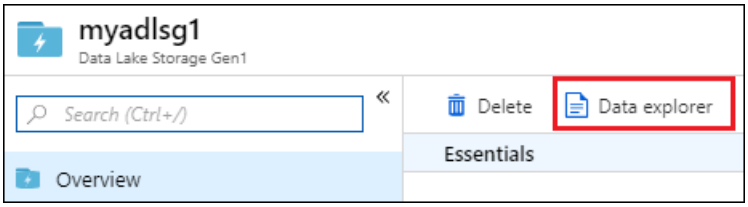
You can also create a Data Lake Storage Gen1 account using Azure Resource Manager templates. These templates are accessible from Azure QuickStart Templates:

- Without data encryption: Deploy Azure Data Lake Storage Gen1 account with no data encryption.
- With data encryption using Data Lake Storage Gen1: Deploy Data Lake Storage Gen1 account with encryption(Data Lake).
- With data encryption using Azure Key Vault: Deploy Data Lake Storage Gen1 account with encryption(Key Vault).

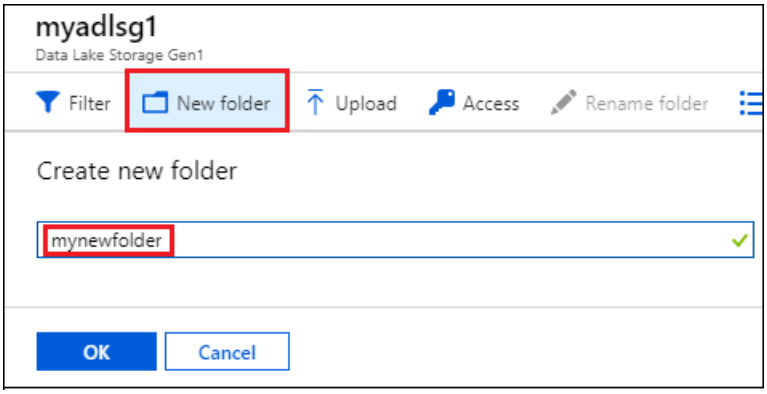
Create folders

You can create folders under your Data Lake Storage Gen1 account to manage and store data.

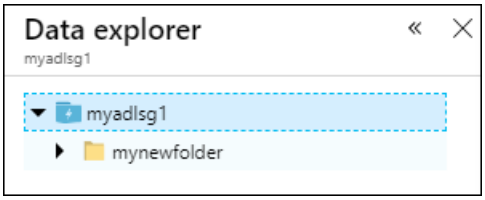
1. Open the Data Lake Storage Gen1 account that you created. From the left pane, click **All resources**, and then from the **All resources** blade, click the account name under which you want to create folders. If you pinned the account to the startboard, click that account tile.
2. In your Data Lake Storage Gen1 account blade, click **Data Explorer**.



3. From Data Explorer blade, click **New Folder**, enter a name for the new folder, and then click **OK**.



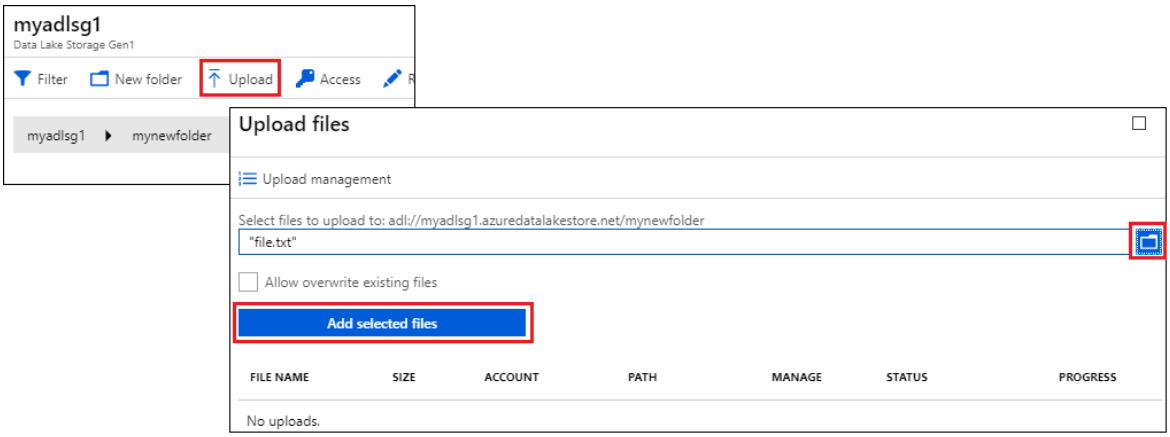
The newly created folder is listed in the **Data Explorer** blade. You can create nested folders up to any level.



Upload data

You can upload your data to a Data Lake Storage Gen1 account directly at the root level or to a folder that you created within the account.

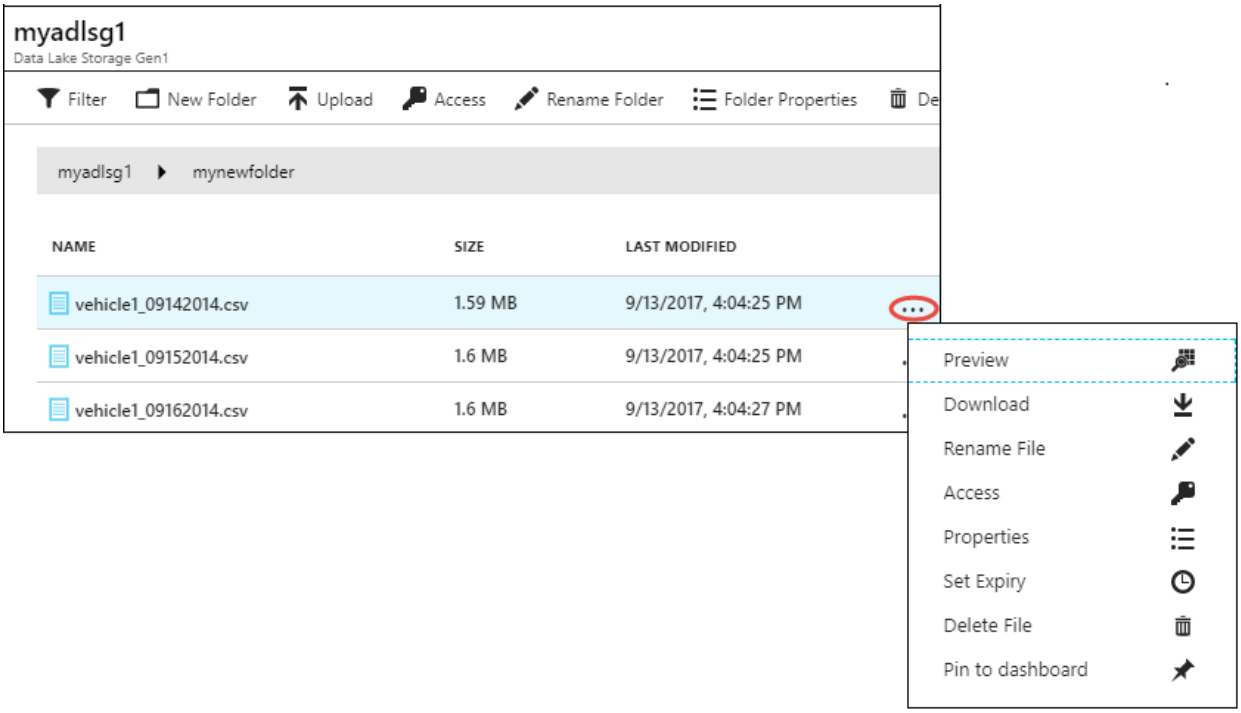
1. From the **Data Explorer** blade, click **Upload**.
2. In the **Upload files** blade, navigate to the files you want to upload, and then click **Add selected files**. You can also select more than one file to upload.



If you are looking for some sample data to upload, you can get the **Ambulance Data** folder from the Azure Data Lake Git Repository.

Actions available on the stored data

Click the ellipsis icon against a file, and from the pop-up menu, click the action you want to perform on the data.

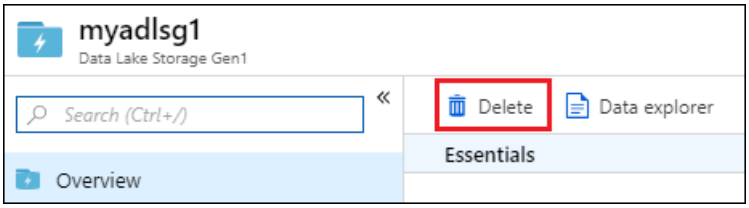


Secure your data

You can secure the data stored in your Data Lake Storage Gen1 account using Azure Active Directory and access control (ACLs). For instructions on how to do that, see Securing data in Azure Data Lake Storage Gen1.

Delete your account

To delete a Data Lake Storage Gen1 account, from your Data Lake Storage Gen1 blade, click **Delete**. To confirm the action, you'll be prompted to enter the name of the account you wish to delete. Enter the name of the account, and then click **Delete**.



Next steps

- Use Azure Data Lake Storage Gen1 for big data requirements
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Get started with Azure Data Lake Storage Gen1 using Azure PowerShell

10/3/2022 • 3 minutes to read • Edit Online

> **NOTE**
>
> On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.
>
> Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

Learn how to use Azure PowerShell to create an Azure Data Lake Storage Gen1 account and perform basic operations such as create folders, upload and download data files, delete your account, etc. For more information about Data Lake Storage Gen1, see Overview of Data Lake Storage Gen1.

Prerequisites

> **NOTE**
>
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

- **An Azure subscription**. See Get Azure free trial.
- **Azure PowerShell 1.0 or greater**. See How to install and configure Azure PowerShell.

Authentication

This article uses a simpler authentication approach with Data Lake Storage Gen1 where you're prompted to enter your Azure account credentials. The access level to Data Lake Storage Gen1 account and file system is then governed by the access level of the logged in user. However, there are other approaches to authenticate with Data Lake Storage Gen1, which are end-user authentication or service-to-service authentication. For instructions and more information on how to authenticate, see End-user authentication or Service-to-service authentication.

Create a Data Lake Storage Gen1 account

1. From your desktop, open a new Windows PowerShell window. Enter the following snippet to log in to your Azure account, set the subscription, and register the Data Lake Storage Gen1 provider. When prompted to log in, make sure you log in as one of the subscription administrators/owner:

```
# Log in to your Azure account
Connect-AzAccount

# List all the subscriptions associated to your account
Get-AzSubscription

# Select a subscription
Set-AzContext -SubscriptionId <subscription ID>

# Register for Azure Data Lake Storage Gen1
Register-AzResourceProvider -ProviderNamespace "Microsoft.DataLakeStore"
```

2. A Data Lake Storage Gen1 account is associated with an Azure resource group. Start by creating a resource group.

```
$resourceGroupName = "<your new resource group name>"
New-AzResourceGroup -Name $resourceGroupName -Location "East US 2"
```

```
ResourceGroupName : mynewresourcegroup
Location          : eastus2
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/                                /resourceGroups/mynewresourcegroup
```

3. Create a Data Lake Storage Gen1 account. The name you specify must only contain lowercase letters and numbers.

```
$dataLakeStorageGen1Name = "<your new Data Lake Storage Gen1 account name>"
New-AzDataLakeStoreAccount -ResourceGroupName $resourceGroupName -Name $dataLakeStorageGen1Name -
Location "East US 2"
```

```
Id                    : /subscriptions/                                /resourceGroups/mynewresourcegroup/pro
                        viders/Microsoft.DataLakeStore/accounts/mynewdatalakestoragegen1
Name                  : mynewdatalakestoragegen1
Type                  : Microsoft.DataLakeStore/accounts
Location              : East US 2
Tags                  : {}
```

4. Verify that the account is successfully created.

```
Test-AzDataLakeStoreAccount -Name $dataLakeStorageGen1Name
```

The output for the cmdlet should be **True**.

Create directory structures

You can create directories under your Data Lake Storage Gen1 account to manage and store data.

1. Specify a root directory.

```
$myrootdir = "/"
```

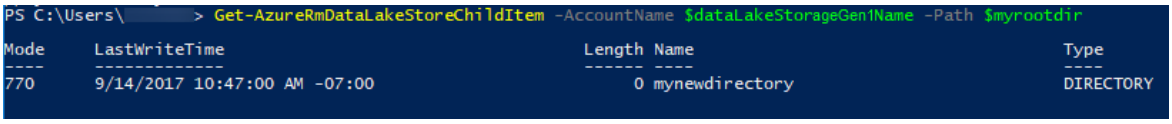
2. Create a new directory called **mynewdirectory** under the specified root.

```
New-AzDataLakeStoreItem -Folder -AccountName $dataLakeStorageGen1Name -Path $myrootdir/mynewdirectory
```

3. Verify that the new directory is successfully created.

```
Get-AzDataLakeStoreChildItem -AccountName $dataLakeStorageGen1Name -Path $myrootdir
```

It should show an output as shown in the following screenshot:



Upload data

You can upload your data to Data Lake Storage Gen1 directly at the root level, or to a directory that you created within the account. The snippets in this section demonstrate how to upload some sample data to the directory (**mynewdirectory**) you created in the previous section.

If you are looking for some sample data to upload, you can get the **Ambulance Data** folder from the Azure Data Lake Git Repository. Download the file and store it in a local directory on your computer, such as C:\sampledata.

```
Import-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name `
   -Path "C:\sampledata\vehicle1_09142014.csv" `
   -Destination $myrootdir\mynewdirectory\vehicle1_09142014.csv
```

Rename, download, and delete data

To rename a file, use the following command:

```
Move-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name `
    -Path $myrootdir\mynewdirectory\vehicle1_09142014.csv `
    -Destination $myrootdir\mynewdirectory\vehicle1_09142014_Copy.csv
```

To download a file, use the following command:

```
Export-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name `
    -Path $myrootdir\mynewdirectory\vehicle1_09142014_Copy.csv `
    -Destination "C:\sampledata\vehicle1_09142014_Copy.csv"
```

To delete a file, use the following command:

```
Remove-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name `
    -Paths $myrootdir\mynewdirectory\vehicle1_09142014_Copy.csv
```

When prompted, enter **Y** to delete the item. If you have more than one file to delete, you can provide all the paths separated by comma.

```
Remove-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name `
    -Paths $myrootdir\mynewdirectory\vehicle1_09142014.csv,
$myrootdir\mynewdirectoryvehicle1_09142014_Copy.csv
```

Delete your account

Use the following command to delete your Data Lake Storage Gen1 account.

```
Remove-AzDataLakeStoreAccount -Name $dataLakeStorageGen1Name
```

When prompted, enter **Y** to delete the account.

Next steps

- Performance tuning guidance for using PowerShell with Azure Data Lake Storage Gen1
- Use Azure Data Lake Storage Gen1 for big data requirements
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Get started with Azure Data Lake Storage Gen1 using the Azure CLI

10/3/2022 • 5 minutes to read • Edit Online

> **NOTE**
>
> On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.
>
> Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

Learn how to use the Azure CLI to create an Azure Data Lake Storage Gen1 account and perform basic operations such as create folders, upload and download data files, delete your account, etc. For more information about Data Lake Storage Gen1, see Overview of Data Lake Storage Gen1.

The Azure CLI is Azure's command-line experience for managing Azure resources. It can be used on macOS, Linux, and Windows. For more information, see Overview of Azure CLI. You can also look at the Azure Data Lake Storage Gen1 CLI reference for a complete list of commands and syntax.

Prerequisites

Before you begin this article, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Azure CLI** - See Install Azure CLI for instructions.

Authentication

This article uses a simpler authentication approach with Data Lake Storage Gen1 where you log in as an end-user user. The access level to the Data Lake Storage Gen1 account and file system is then governed by the access level of the logged in user. However, there are other approaches as well to authenticate with Data Lake Storage Gen1, which are **end-user authentication** or **service-to-service authentication**. For instructions and more information on how to authenticate, see End-user authentication or Service-to-service authentication.

Log in to your Azure subscription

1. Log into your Azure subscription.

   ```
   az login
   ```

   You get a code to use in the next step. Use a web browser to open the page https://aka.ms/devicelogin and enter the code to authenticate. You are prompted to log in using your credentials.

2. Once you log in, the window lists all the Azure subscriptions that are associated with your account. Use the following command to use a specific subscription.

   ```
   az account set --subscription <subscription id>
   ```

Create an Azure Data Lake Storage Gen1 account

1. Create a new resource group. In the following command, provide the parameter values you want to use. If the location name contains spaces, put it in quotes. For example "East US 2".

   ```
   az group create --location "East US 2" --name myresourcegroup
   ```

2. Create the Data Lake Storage Gen1 account.

   ```
   az dls account create --account mydatalakestoragegen1 --resource-group myresourcegroup
   ```

Create folders in a Data Lake Storage Gen1 account

You can create folders under your Azure Data Lake Storage Gen1 account to manage and store data. Use the following command to create a folder called **mynewfolder** at the root of the Data Lake Storage Gen1 account.

```
az dls fs create --account mydatalakestoragegen1 --path /mynewfolder --folder
```

> **NOTE**
> The `--folder` parameter ensures that the command creates a folder. If this parameter is not present, the command creates an empty file called mynewfolder at the root of the Data Lake Storage Gen1 account.

Upload data to a Data Lake Storage Gen1 account

You can upload data to Data Lake Storage Gen1 directly at the root level or to a folder that you created within the account. The snippets below demonstrate how to upload some sample data to the folder (**mynewfolder**) you created in the previous section.

If you are looking for some sample data to upload, you can get the **Ambulance Data** folder from the Azure Data Lake Git Repository. Download the file and store it in a local directory on your computer, such as C:\sampledata.

```
az dls fs upload --account mydatalakestoragegen1 --source-path
"C:\SampleData\AmbulanceData\vehicle1_09142014.csv" --destination-path "/mynewfolder/vehicle1_09142014.csv"
```

> **NOTE**
> For the destination, you must specify the complete path including the file name.

List files in a Data Lake Storage Gen1 account

Use the following command to list the files in a Data Lake Storage Gen1 account.

```
az dls fs list --account mydatalakestoragegen1 --path /mynewfolder
```

The output of this should be similar to the following:

```
[
 {
  "accessTime": 1491323529542,
  "aclBit": false,
  "blockSize": 268435456,
  "group": "1808bd5f-62af-45f4-89d8-03c5e81bac20",
  "length": 1589881,
  "modificationTime": 1491323531638,
  "msExpirationTime": 0,
  "name": "mynewfolder/vehicle1_09142014.csv",
  "owner": "1808bd5f-62af-45f4-89d8-03c5e81bac20",
  "pathSuffix": "vehicle1_09142014.csv",
  "permission": "770",
  "replication": 1,
  "type": "FILE"
 }
]
```


Rename, download, and delete data from a Data Lake Storage Gen1 account

- **To rename a file**, use the following command:

  ```
  az dls fs move --account mydatalakestoragegen1 --source-path /mynewfolder/vehicle1_09142014.csv --
  destination-path /mynewfolder/vehicle1_09142014_copy.csv
  ```

- **To download a file**, use the following command. Make sure the destination path you specify already exists.

  ```
  az dls fs download --account mydatalakestoragegen1 --source-path
  /mynewfolder/vehicle1_09142014_copy.csv --destination-path
  "C:\mysampledata\vehicle1_09142014_copy.csv"
  ```

  > **NOTE**
  > The command creates the destination folder if it does not exist.

- **To delete a file**, use the following command:

  ```
  az dls fs delete --account mydatalakestoragegen1 --path /mynewfolder/vehicle1_09142014_copy.csv
  ```

  If you want to delete the folder **mynewfolder** and the file **vehicle1_09142014_copy.csv** together in one command, use the --recurse parameter

  ```
  az dls fs delete --account mydatalakestoragegen1 --path /mynewfolder --recurse
  ```

Work with permissions and ACLs for a Data Lake Storage Gen1 account

In this section you learn about how to manage ACLs and permissions using the Azure CLI. For a detailed discussion on how ACLs are implemented in Azure Data Lake Storage Gen1, see Access control in Azure Data Lake Storage Gen1.

- **To update the owner of a file/folder**, use the following command:

```
  az dls fs access set-owner --account mydatalakestoragegen1 --path
/mynewfolder/vehicle1_09142014.csv --group 80a3ed5f-959e-4696-ba3c-d3c8b2db6766 --owner 6361e05d-
c381-4275-a932-5535806bb323
```

- **To update the permissions for a file/folder**, use the following command:

```
  az dls fs access set-permission --account mydatalakestoragegen1 --path
/mynewfolder/vehicle1_09142014.csv --permission 777
```

- **To get the ACLs for a given path**, use the following command:

```
  az dls fs access show --account mydatalakestoragegen1 --path /mynewfolder/vehicle1_09142014.csv
```

  The output should be similar to the following:

```
  {
   "entries": [
   "user::rwx",
   "group::rwx",
   "other::---"
   ],
   "group": "1808bd5f-62af-45f4-89d8-03c5e81bac20",
   "owner": "1808bd5f-62af-45f4-89d8-03c5e81bac20",
   "permission": "770",
   "stickyBit": false
  }
```

- **To set an entry for an ACL**, use the following command:

```
  az dls fs access set-entry --account mydatalakestoragegen1 --path /mynewfolder --acl-spec
user:6360e05d-c381-4275-a932-5535806bb323:-w-
```

- **To remove an entry for an ACL**, use the following command:

```
  az dls fs access remove-entry --account mydatalakestoragegen1 --path /mynewfolder --acl-spec
user:6360e05d-c381-4275-a932-5535806bb323
```

- **To remove an entire default ACL**, use the following command:

```
  az dls fs access remove-all --account mydatalakestoragegen1 --path /mynewfolder --default-acl
```

- **To remove an entire non-default ACL**, use the following command:

```
  az dls fs access remove-all --account mydatalakestoragegen1 --path /mynewfolder
```

Delete a Data Lake Storage Gen1 account

Use the following command to delete a Data Lake Storage Gen1 account.

```
az dls account delete --account mydatalakestoragegen1
```

When prompted, enter **Y** to delete the account.

Next steps

- Use Azure Data Lake Storage Gen1 for big data requirements
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Azure Policy Regulatory Compliance controls for Azure Data Lake Storage Gen1

10/3/2022 • 6 minutes to read • Edit Online

Regulatory Compliance in Azure Policy provides Microsoft created and managed initiative definitions, known as *built-ins*, for the **compliance domains** and **security controls** related to different compliance standards. This page lists the **compliance domains** and **security controls** for Azure Data Lake Storage Gen1. You can assign the built-ins for a **security control** individually to help make your Azure resources compliant with the specific standard.

The title of each built-in policy definition links to the policy definition in the Azure portal. Use the link in the **Policy Version** column to view the source on the Azure Policy GitHub repo.

> **IMPORTANT**
> Each control is associated with one or more Azure Policy definitions. These policies might help you assess compliance with the control. However, there often isn't a one-to-one or complete match between a control and one or more policies. As such, **Compliant** in Azure Policy refers only to the policies themselves. This doesn't ensure that you're fully compliant with all requirements of a control. In addition, the compliance standard includes controls that aren't addressed by any Azure Policy definitions at this time. Therefore, compliance in Azure Policy is only a partial view of your overall compliance status. The associations between controls and Azure Policy Regulatory Compliance definitions for these compliance standards can change over time.

Azure Security Benchmark

The Azure Security Benchmark provides recommendations on how you can secure your cloud solutions on Azure. To see how this service completely maps to the Azure Security Benchmark, see the Azure Security Benchmark mapping files.

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - Azure Security Benchmark.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Logging and Threat Detection	LT-3	Enable logging for security investigation	Resource logs in Azure Data Lake Store should be enabled	5.0.0

Azure Security Benchmark v1

The Azure Security Benchmark provides recommendations on how you can secure your cloud solutions on Azure. To see how this service completely maps to the Azure Security Benchmark, see the Azure Security Benchmark mapping files.

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - Azure Security Benchmark.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Logging and Monitoring	2.3	Enable audit logging for Azure resources	Resource logs in Azure Data Lake Store should be enabled	5.0.0

CIS Microsoft Azure Foundations Benchmark 1.3.0

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - CIS Microsoft Azure Foundations Benchmark 1.3.0. For more information about this compliance standard, see CIS Microsoft Azure Foundations Benchmark.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
5 Logging and Monitoring	CIS Microsoft Azure Foundations Benchmark recommendation 5.3	Ensure that Diagnostic Logs are enabled for all services which support it.	Resource logs in Azure Data Lake Store should be enabled	5.0.0

CMMC Level 3

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - CMMC Level 3. For more information about this compliance standard, see Cybersecurity Maturity Model Certification (CMMC).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC.3.177	Employ FIPS-validated cryptography when used to protect the confidentiality of CUI.	Require encryption on Data Lake Store accounts	1.0.0
System and Communications Protection	SC.3.191	Protect the confidentiality of CUI at rest.	Require encryption on Data Lake Store accounts	1.0.0

FedRAMP High

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - FedRAMP High. For more information about this compliance standard, see FedRAMP High.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Audit And Accountability	AU-6 (4)	Central Review And Analysis	Resource logs in Azure Data Lake Store should be enabled	5.0.0

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY	POLICY VERSION
Audit And Accountability	AU-6 (5)	Integration / Scanning And Monitoring Capabilities	Resource logs in Azure Data Lake Store should be enabled	5.0.0
Audit And Accountability	AU-12	Audit Generation	Resource logs in Azure Data Lake Store should be enabled	5.0.0
Audit And Accountability	AU-12 (1)	System-Wide / Time-Correlated Audit Trail	Resource logs in Azure Data Lake Store should be enabled	5.0.0

FedRAMP Moderate

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - FedRAMP Moderate. For more information about this compliance standard, see FedRAMP Moderate.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Audit And Accountability	AU-12	Audit Generation	Resource logs in Azure Data Lake Store should be enabled	5.0.0

HIPAA HITRUST 9.2

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - HIPAA HITRUST 9.2. For more information about this compliance standard, see HIPAA HITRUST 9.2.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
03 Portable Media Security	0304.09o3Organizational.1-09.o	09.07 Media Handling	Require encryption on Data Lake Store accounts	1.0.0
12 Audit Logging & Monitoring	1202.09aa1System.1-09.aa	09.10 Monitoring	Resource logs in Azure Data Lake Store should be enabled	5.0.0

New Zealand ISM Restricted

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - New Zealand ISM Restricted. For more information about this compliance standard, see New Zealand ISM Restricted.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Access Control and Passwords	AC-17	16.6.9 Events to be logged	Resource logs in Azure Data Lake Store should be enabled	5.0.0

NIST SP 800-53 Rev. 5

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - NIST SP 800-53 Rev. 5. For more information about this compliance standard, see NIST SP 800-53 Rev. 5.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Audit and Accountability	AU-6 (4)	Central Review and Analysis	Resource logs in Azure Data Lake Store should be enabled	5.0.0
Audit and Accountability	AU-6 (5)	Integrated Analysis of Audit Records	Resource logs in Azure Data Lake Store should be enabled	5.0.0
Audit and Accountability	AU-12	Audit Record Generation	Resource logs in Azure Data Lake Store should be enabled	5.0.0
Audit and Accountability	AU-12 (1)	System-wide and Time-correlated Audit Trail	Resource logs in Azure Data Lake Store should be enabled	5.0.0

NZ ISM Restricted v3.5

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - NZ ISM Restricted v3.5. For more information about this compliance standard, see NZ ISM Restricted v3.5.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Access Control and Passwords	NZISM Security Benchmark AC-18	16.6.9 Events to be logged	Resource logs in Azure Data Lake Store should be enabled	5.0.0

RMIT Malaysia

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see Azure Policy Regulatory Compliance - RMIT Malaysia. For more information about this compliance standard, see RMIT Malaysia.

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Security of Digital Services	RMiT 10.66	Security of Digital Services - 10.66	Deploy Diagnostic Settings for Data Lake Storage Gen1 to Event Hub	2.0.0
Security of Digital Services	RMiT 10.66	Security of Digital Services - 10.66	Deploy Diagnostic Settings for Data Lake Storage Gen1 to Log Analytics workspace	1.0.0

Next steps

- Learn more about Azure Policy Regulatory Compliance.
- See the built-ins on the Azure Policy GitHub repo.

Load data into Azure Data Lake Storage Gen1 by using Azure Data Factory

10/3/2022 • 4 minutes to read • Edit Online

APPLIES TO: Azure Data Factory Azure Synapse Analytics

Azure Data Lake Storage Gen1 (previously known as Azure Data Lake Store) is an enterprise-wide hyper-scale repository for big data analytic workloads. Data Lake Storage Gen1 lets you capture data of any size, type, and ingestion speed. The data is captured in a single place for operational and exploratory analytics.

Azure Data Factory is a fully managed cloud-based data integration service. You can use the service to populate the lake with data from your existing system and save time when building your analytics solutions.

Azure Data Factory offers the following benefits for loading data into Data Lake Storage Gen1:

- **Easy to set up**: An intuitive 5-step wizard with no scripting required.
- **Rich data store support**: Built-in support for a rich set of on-premises and cloud-based data stores. For a detailed list, see the table of Supported data stores.
- **Secure and compliant**: Data is transferred over HTTPS or ExpressRoute. The global service presence ensures that your data never leaves the geographical boundary.
- **High performance**: Up to 1-GB/s data loading speed into Data Lake Storage Gen1. For details, see Copy activity performance.

This article shows you how to use the Data Factory Copy Data tool to *load data from Amazon S3 into Data Lake Storage Gen1*. You can follow similar steps to copy data from other types of data stores.

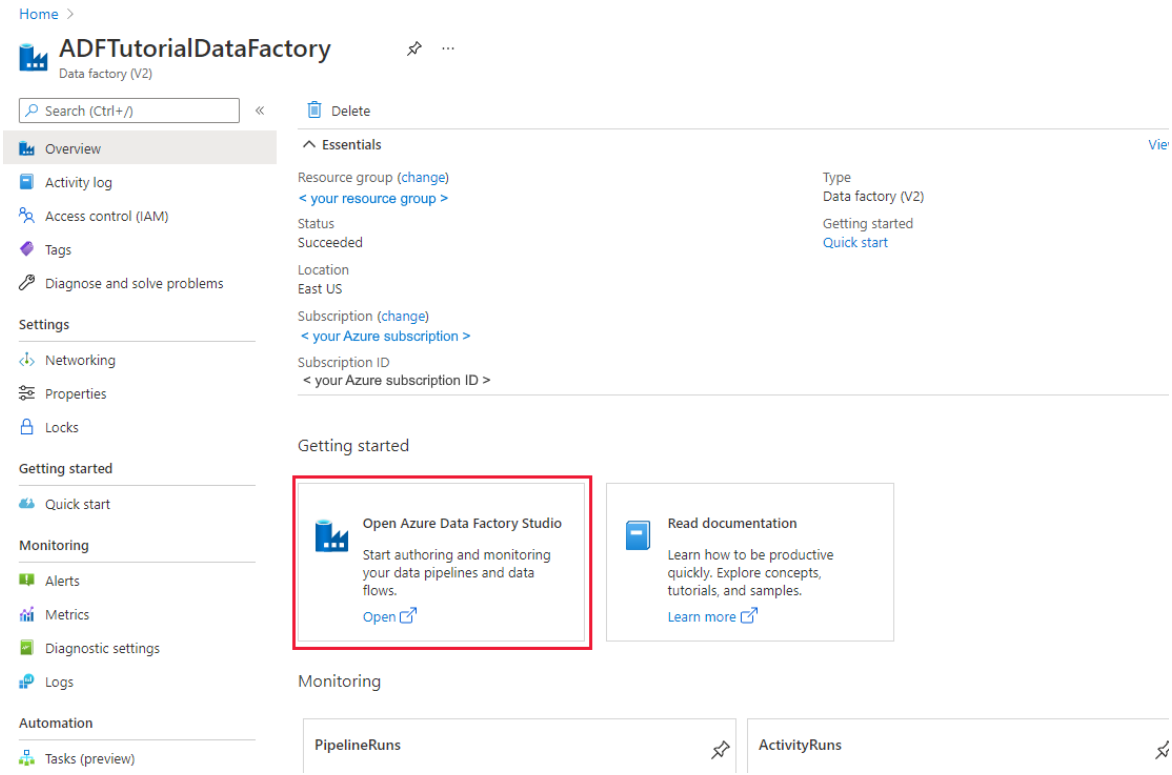
> **NOTE**
> For more information, see Copy data to or from Data Lake Storage Gen1 by using Azure Data Factory.

Prerequisites

- Azure subscription: If you don't have an Azure subscription, create a free account before you begin.
- Data Lake Storage Gen1 account: If you don't have a Data Lake Storage Gen1 account, see the instructions in Create a Data Lake Storage Gen1 account.
- Amazon S3: This article shows how to copy data from Amazon S3. You can use other data stores by following similar steps.

Create a data factory

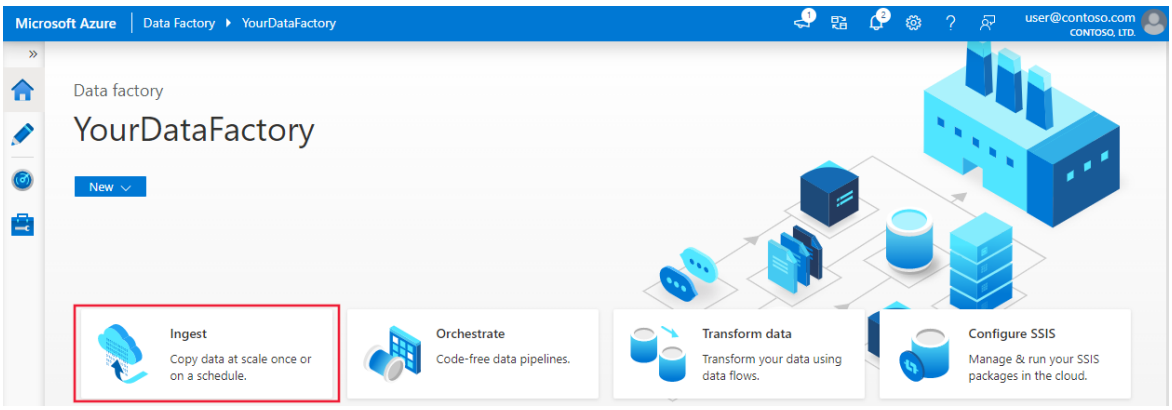
1. If you have not created your data factory yet, follow the steps in Quickstart: Create a data factory by using the Azure portal and Azure Data Factory Studio to create one. After creating it, browse to the data factory in the Azure portal.



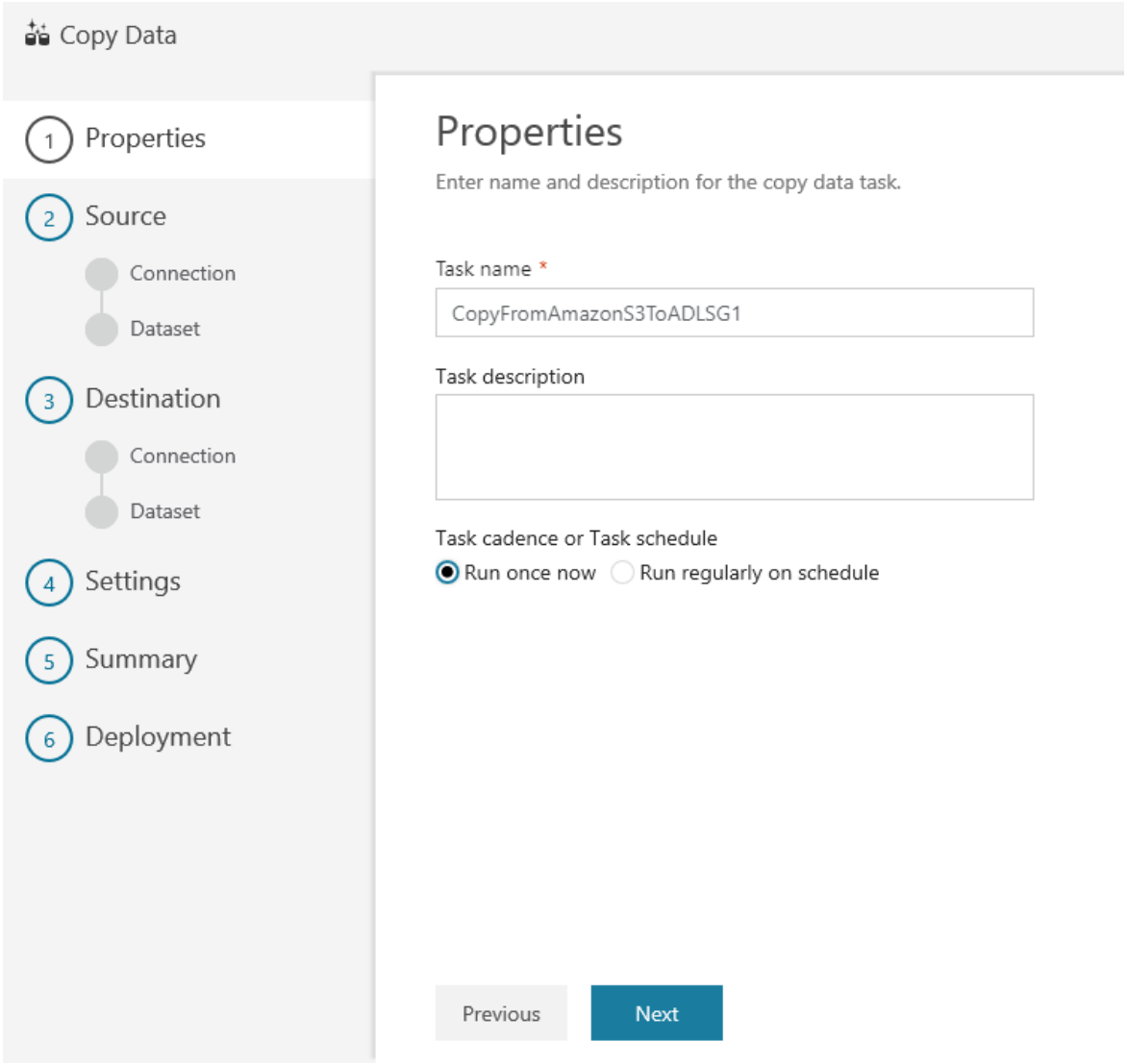
2. Select **Open** on the **Open Azure Data Factory Studio** tile to launch the Data Integration application in a separate tab.

Load data into Data Lake Storage Gen1

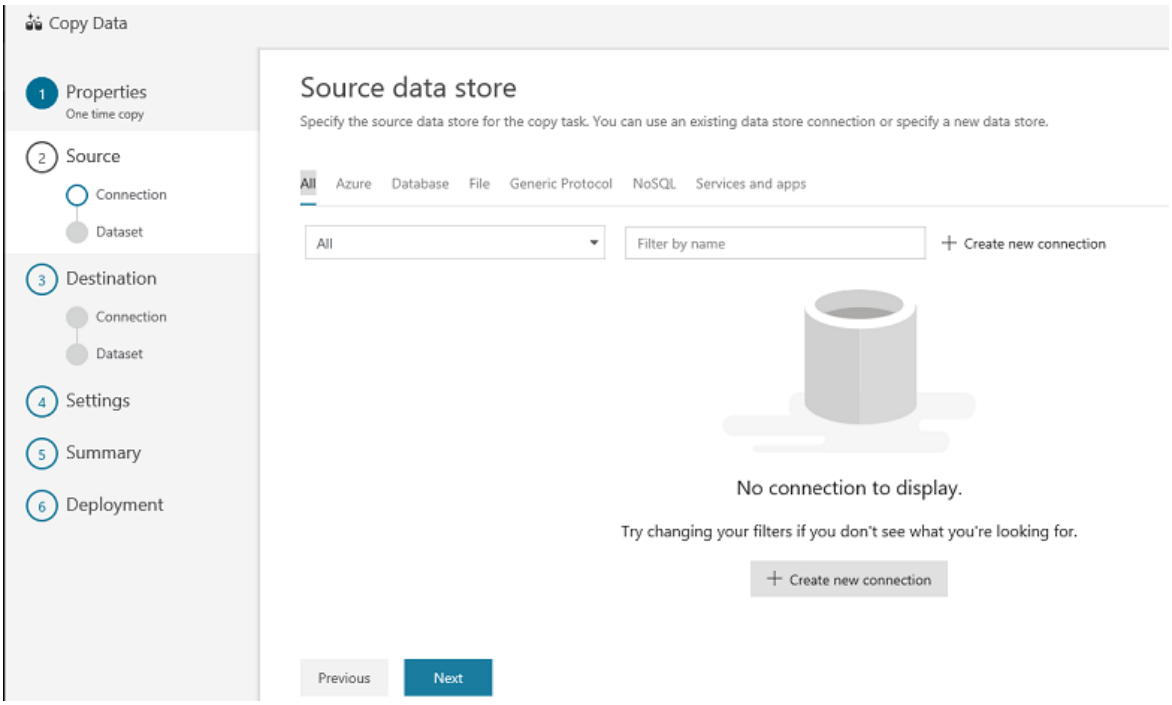
1. In the home page, select the **Ingest** tile to launch the Copy Data tool:



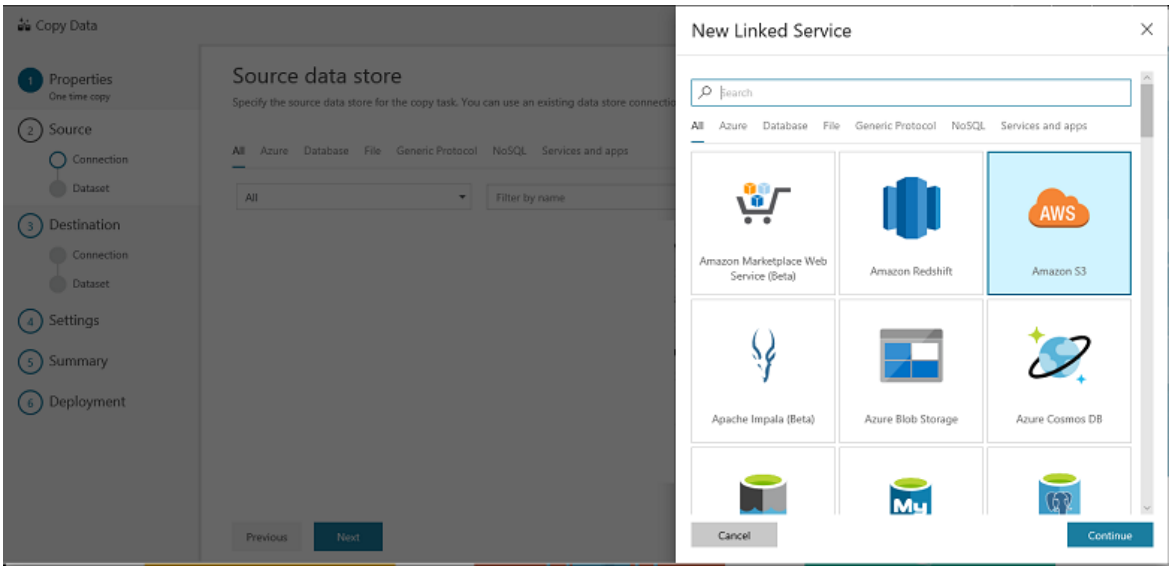
2. In the **Properties** page, specify **CopyFromAmazonS3ToADLS** for the **Task name** field, and select **Next**:



3. In the **Source data store** page, select **+ Create new connection**:



Select **Amazon S3**, and select **Continue**

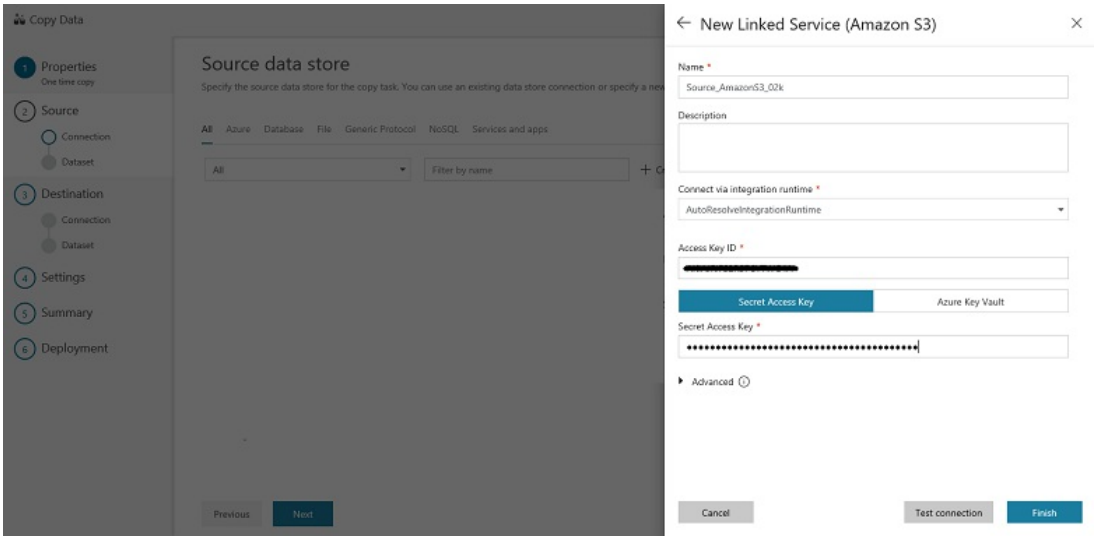


4. In the **Specify Amazon S3 connection** page, do the following steps:

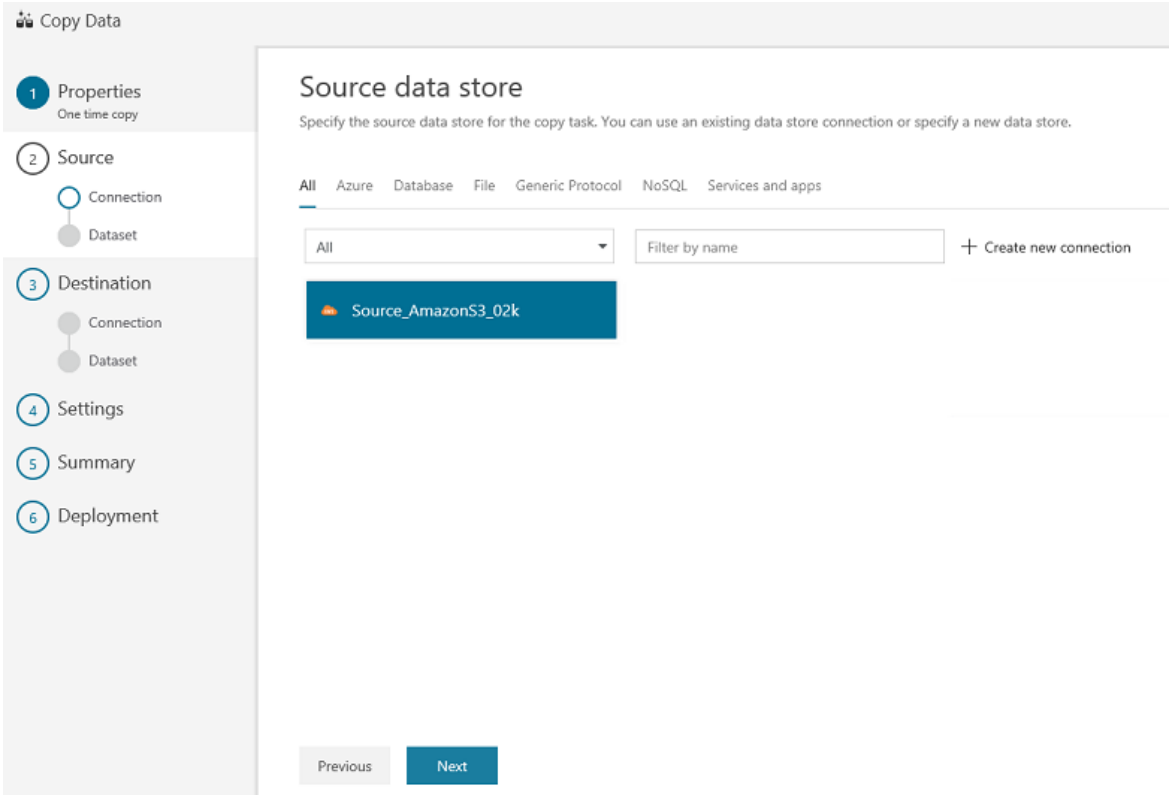
   a. Specify the **Access Key ID** value.

   b. Specify the **Secret Access Key** value.

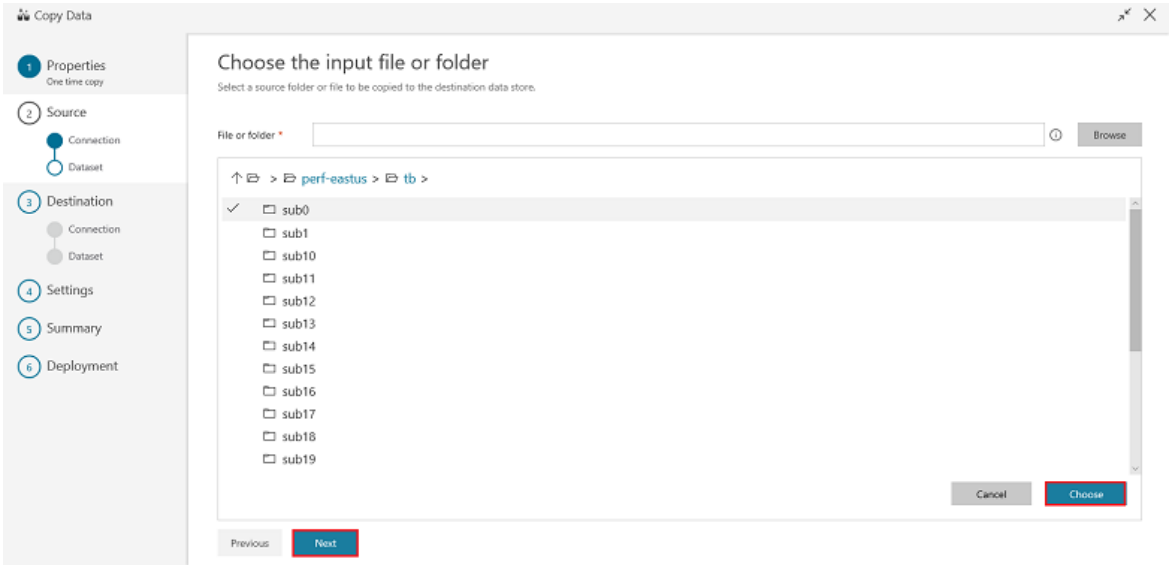
   c. Select **Finish**.



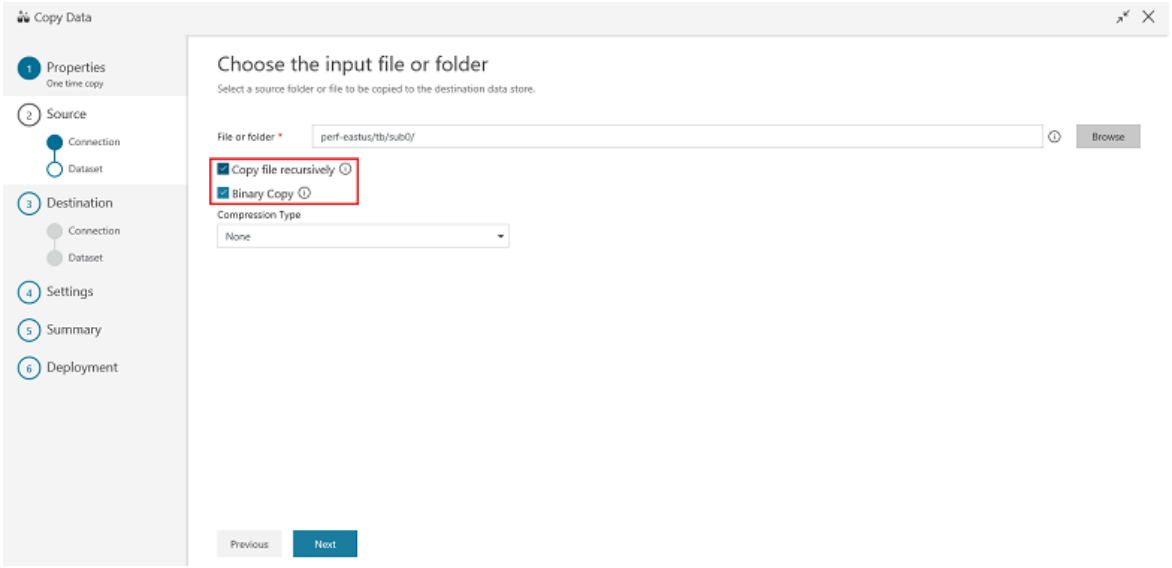
   d. You will see a new connection. Select **Next**.



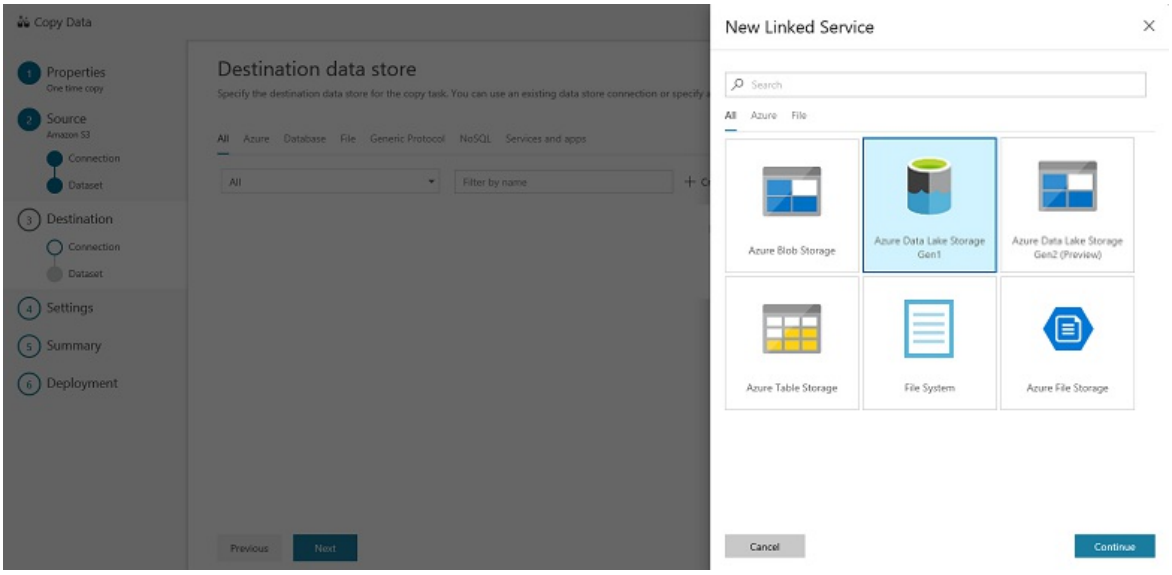
5. In the **Choose the input file or folder** page, browse to the folder and file that you want to copy over. Select the folder/file, select **Choose**, and then select **Next**:



6. Choose the copy behavior by selecting the **Copy files recursively** and **Binary copy** (copy files as-is) options. Select **Next**:



7. In the **Destination data store** page, select **+ Create new connection**, and then select **Azure Data Lake Storage Gen1**, and select **Continue**:



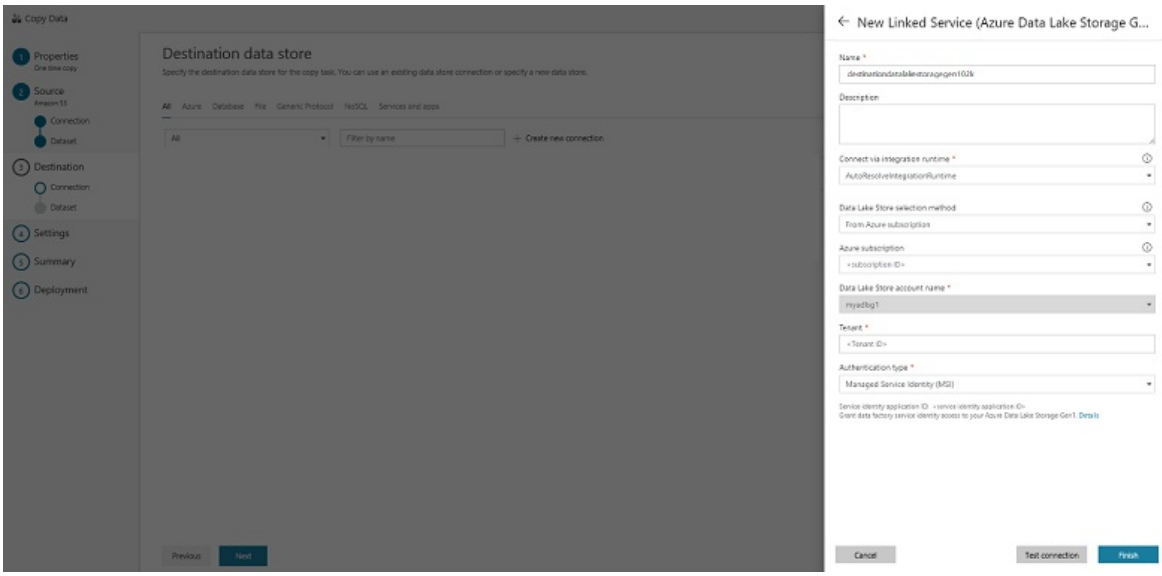
8. In the **New Linked Service (Azure Data Lake Storage Gen1)** page, do the following steps:

   a. Select your Data Lake Storage Gen1 account for the **Data Lake Store account name**.

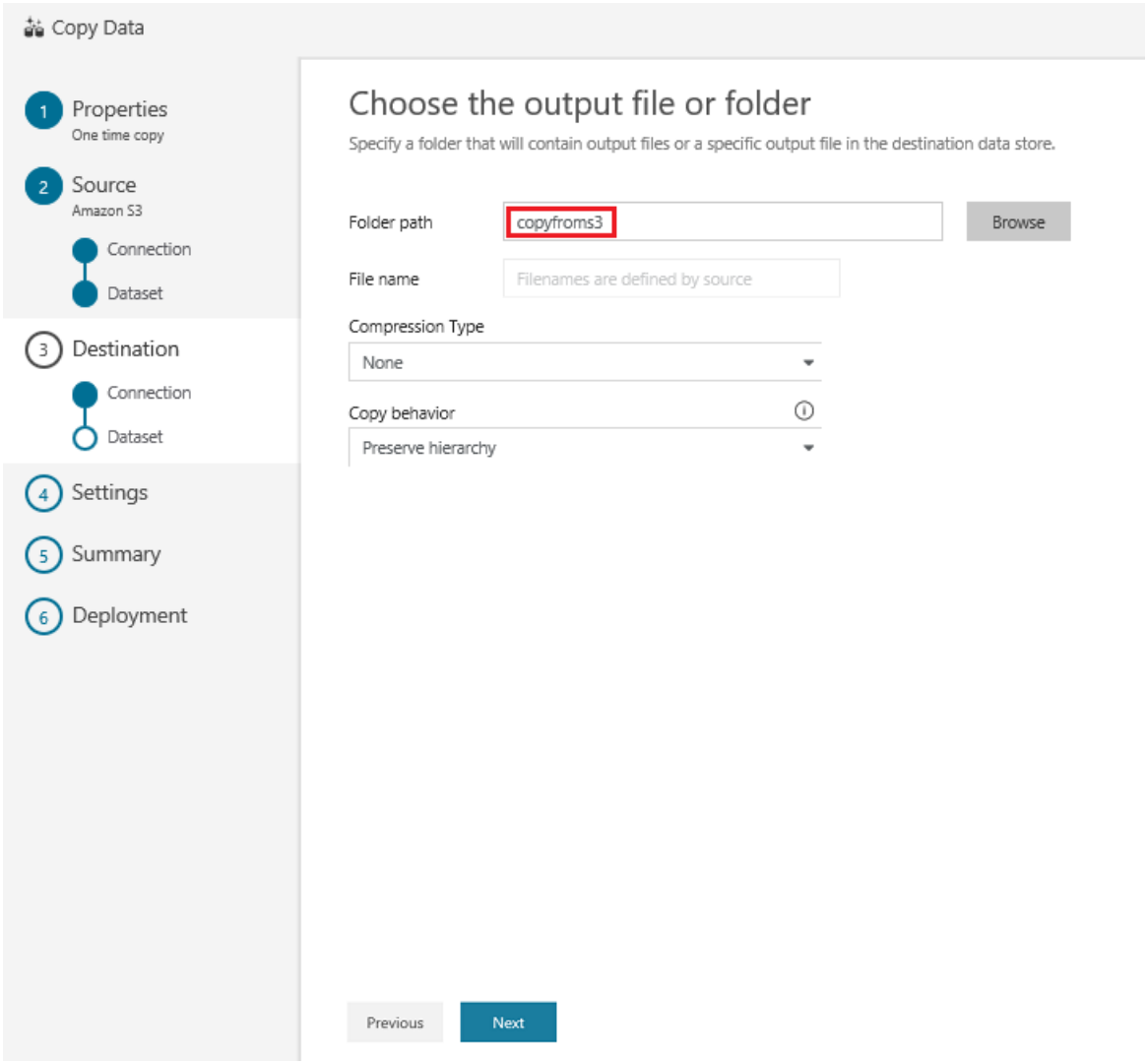
   b. Specify the **Tenant**, and select Finish.

   c. Select **Next**.

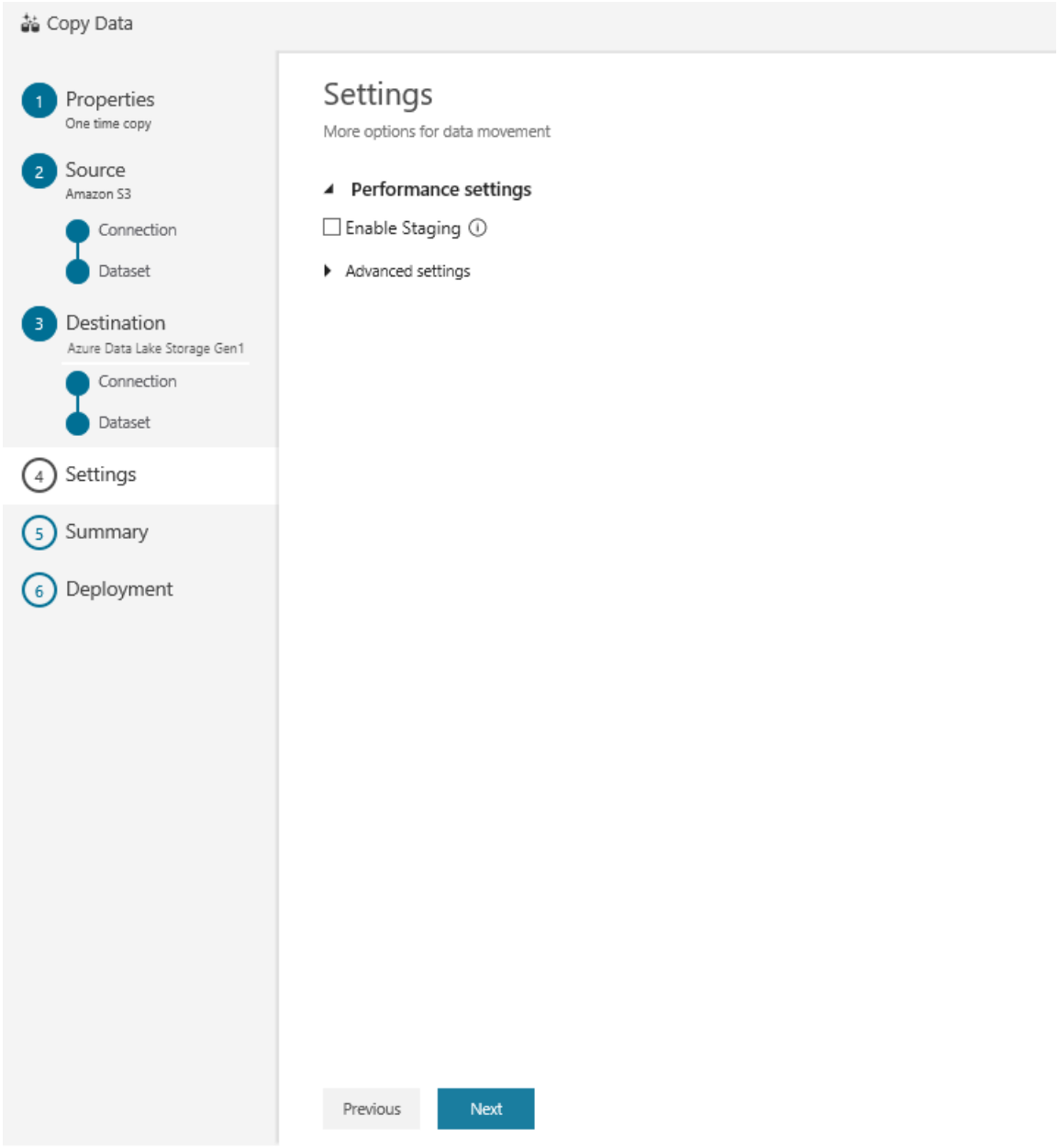
> **IMPORTANT**
>
> In this walkthrough, you use a managed identity for Azure resources to authenticate your Data Lake Storage Gen1 account. Be sure to grant the MSI the proper permissions in Data Lake Storage Gen1 by following these instructions.



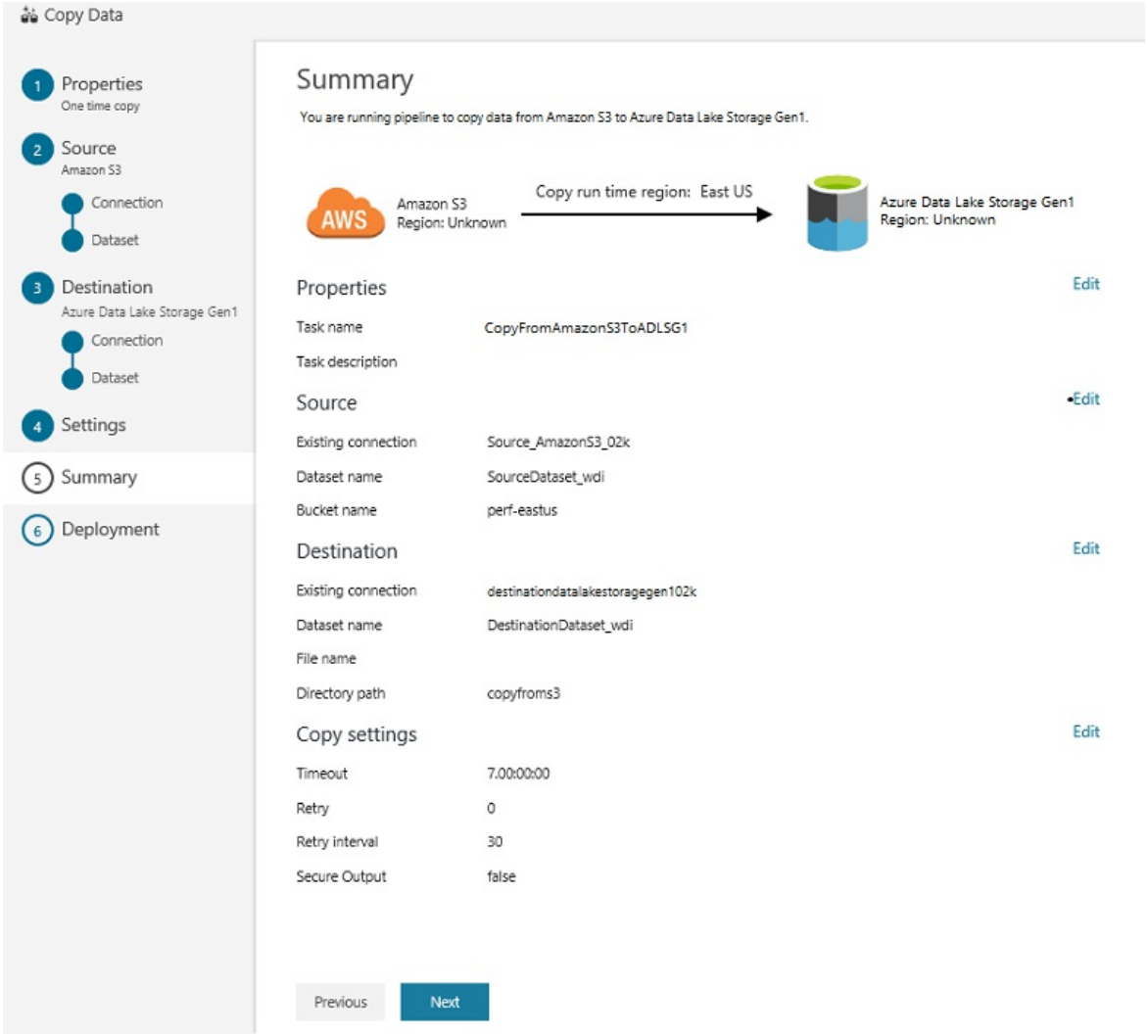
9. In the **Choose the output file or folder** page, enter **copyfroms3** as the output folder name, and select **Next**:



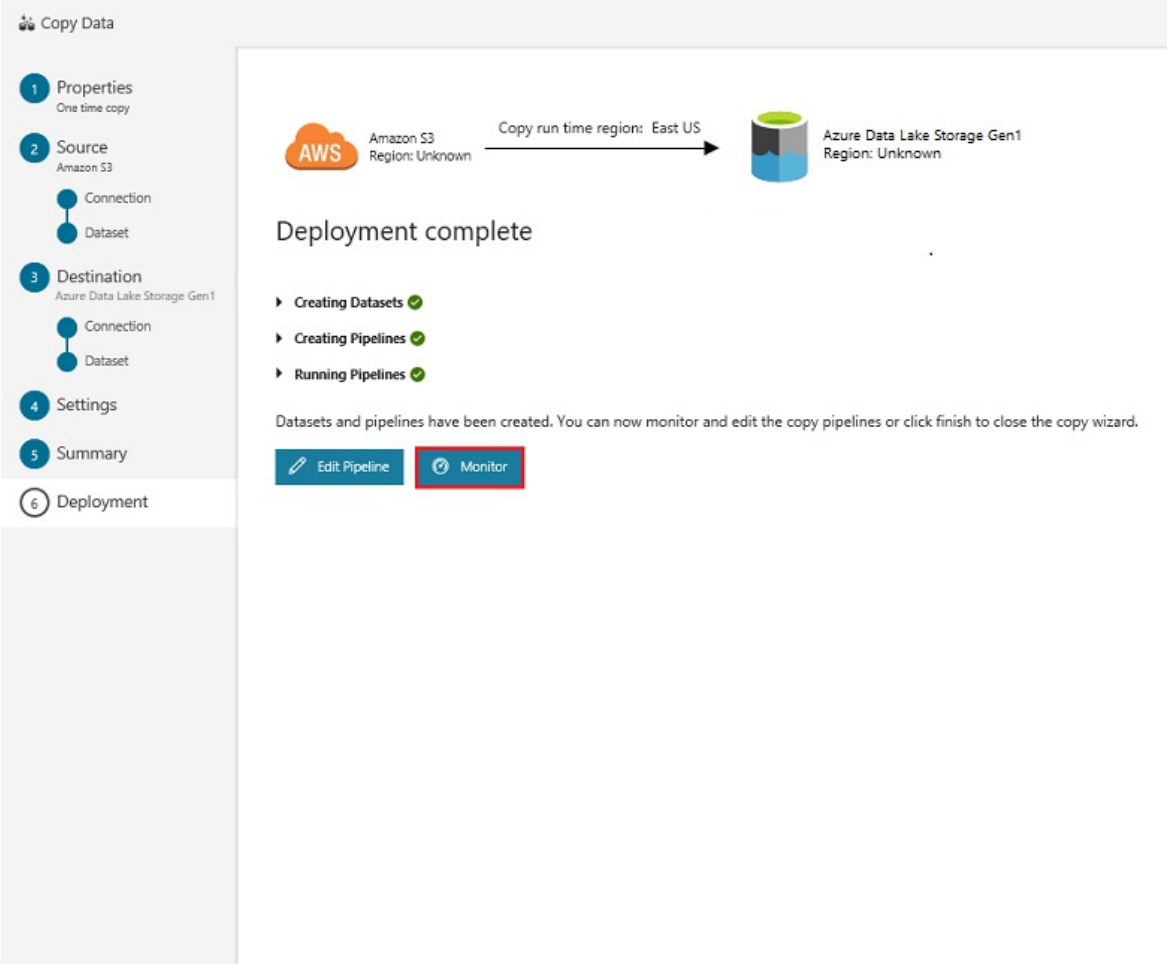
10. In the **Settings** page, select **Next**:



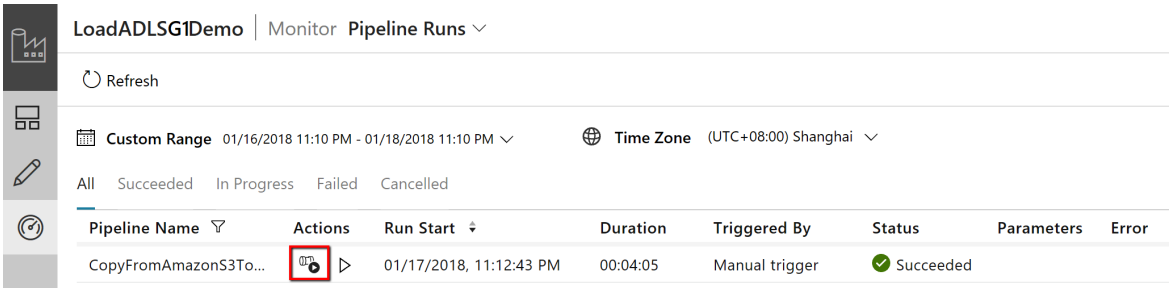
11. In the **Summary** page, review the settings, and select **Next**:



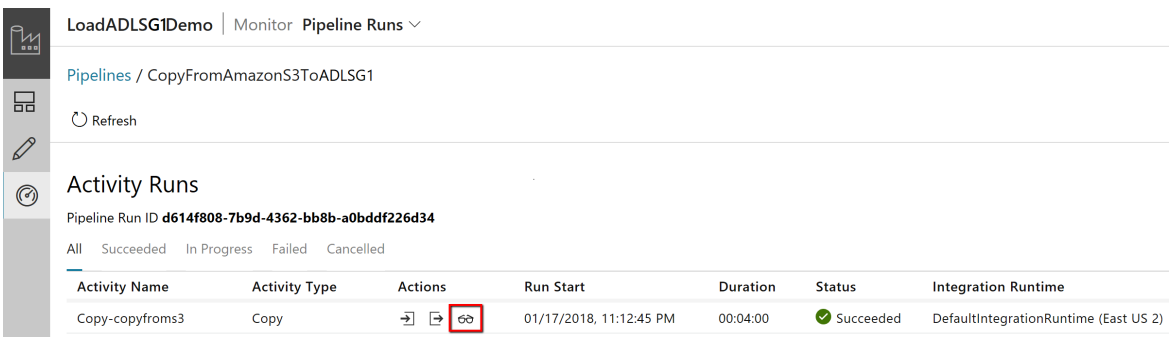
12. In the **Deployment page**, select **Monitor** to monitor the pipeline (task):



13. Notice that the **Monitor** tab on the left is automatically selected. The **Actions** column includes links to view activity run details and to rerun the pipeline:



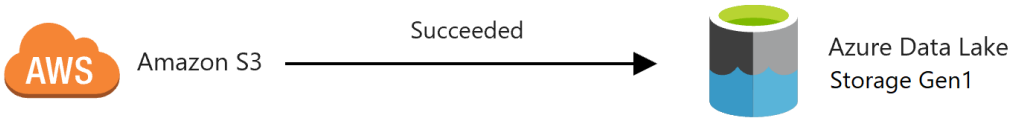
14. To view activity runs that are associated with the pipeline run, select the **View Activity Runs** link in the **Actions** column. There's only one activity (copy activity) in the pipeline, so you see only one entry. To switch back to the pipeline runs view, select the **Pipelines** link at the top. Select **Refresh** to refresh the list.



15. To monitor the execution details for each copy activity, select the **Details** link under **Actions** in the activity monitoring view. You can monitor details like the volume of data copied from the source to the sink, data throughput, execution steps with corresponding duration, and used configurations:

Details

Learn more on copy performance details from here.



Data read: 107.281 GB
Files read: 10

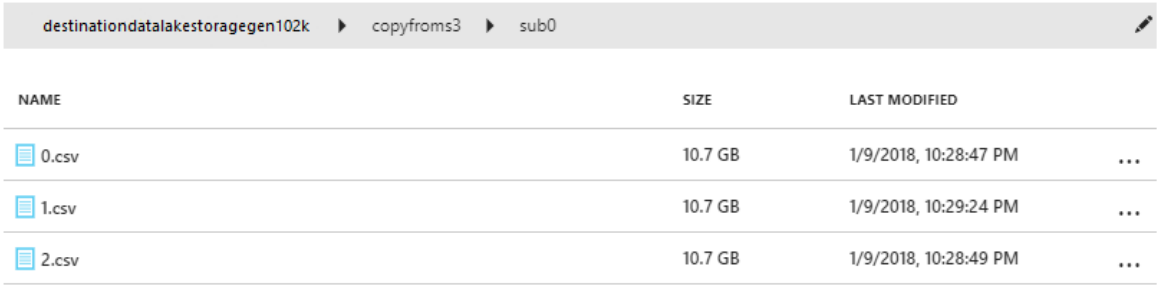
Data written: 107.281 GB
Files written: 10
Throughput: 467.707 MB/s

Execution details:

Amazon S3 → Azure Data Lake Storage Gen1 Queue 00:00:02 | Transfer 00:03:39

Start time	01/17/2018, 11:13:00 PM
Duration	00:03:41
Used DMUs	32
Used parallel copies	8

16. Verify that the data is copied into your Data Lake Storage Gen1 account:



Next steps

Advance to the following article to learn about Data Lake Storage Gen1 support:

Azure Data Lake Storage Gen1 connector

Manage Data Lake Storage Gen1 resources by using Storage Explorer

10/3/2022 • 6 minutes to read • Edit Online

Azure Data Lake Storage Gen1 is a service for storing large amounts of unstructured data, such as text or binary data. You can get access to the data from anywhere via HTTP or HTTPS. Data Lake Storage Gen1 in Azure Storage Explorer enables you to access and manage Data Lake Storage Gen1 data and resources, along with other Azure entities like blobs and queues. Now you can use the same tool to manage your different Azure entities in one place.

Another advantage is that you don't need to have subscription permission to manage Data Lake Storage Gen1 data. In Storage Explorer, you can attach the Data Lake Storage Gen1 path to the **Local & Attached** node as long as someone grants the permission.

Prerequisites

To complete the steps in this article, you need the following prerequisites:

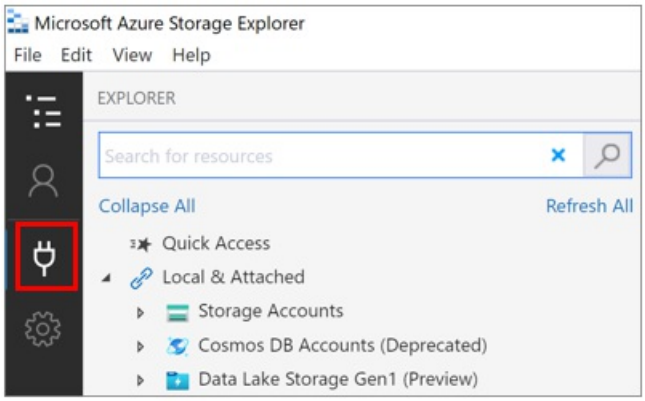
- An Azure subscription. See Get Azure free trial.
- A Data Lake Storage Gen1 account. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1.

Install Storage Explorer

Install the latest Azure Storage Explorer bits from the product webpage. The installation supports Windows, Linux, and Mac versions.

Connect to an Azure subscription

1. In Storage Explorer, select the plug-in icon.



   This opens the **Connect to Azure Storage** dialog box.

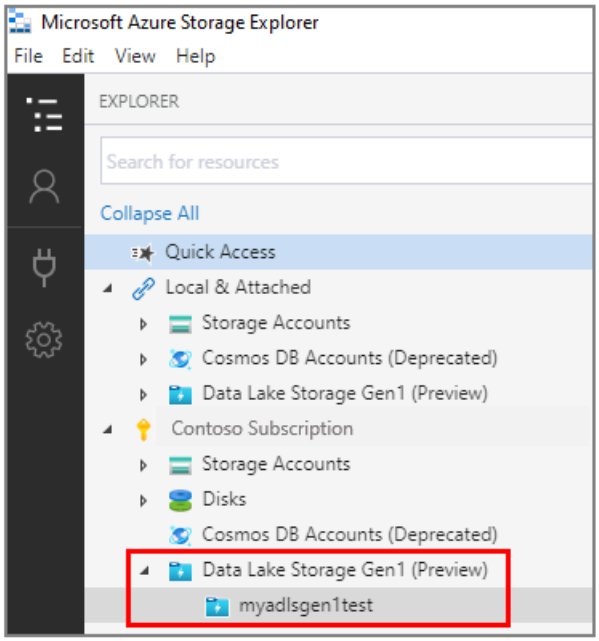
2. On the **Select Resource** page, select **Subscription**.

3. On the **Select Azure Environment** page, select the Azure environment to sign in to, and then select **Next**.

4. In the **Sign in** dialog box, enter your Azure credentials, and then select **Next**.

5. In Storage Explorer, in the **ACCOUNT MANAGEMENT** pane, select the subscription that contains the Data Lake Storage Gen1 account that you want to manage, and then select **Open Explorer**.

6. In the **EXPLORER** pane, expand your subscription. The pane updates and displays the accounts in the selected subscription. This includes any Data Lake Storage Gen1 accounts, for example:



Connect to Data Lake Storage Gen1

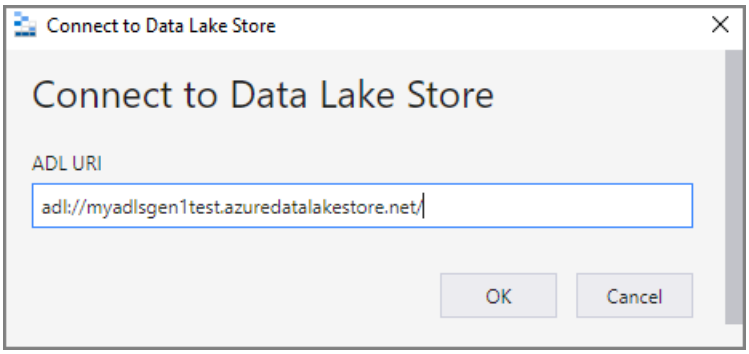
You can access resources that don't exist in your subscription if someone gives you the URI for the resources. You can then connect to Data Lake Storage Gen1 by using the URI after you sign in.

1. Open Storage Explorer.

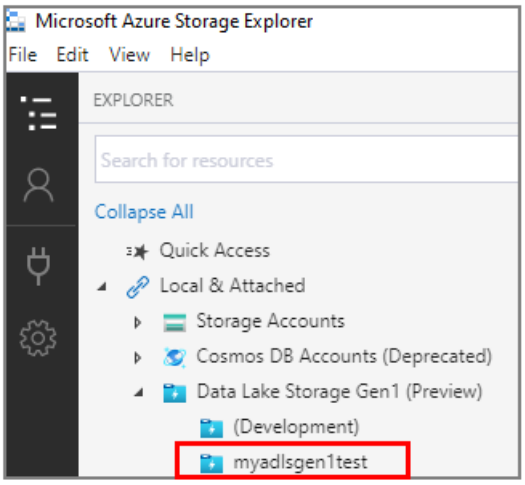
2. Expand **Local & Attached**.

3. Right-click **Data Lake Storage Gen1 (Preview)**, and then select **Connect to Data Lake Storage Gen1**.

4. Enter the URI, for example:



The tool browses to the location of the URL that you just entered.

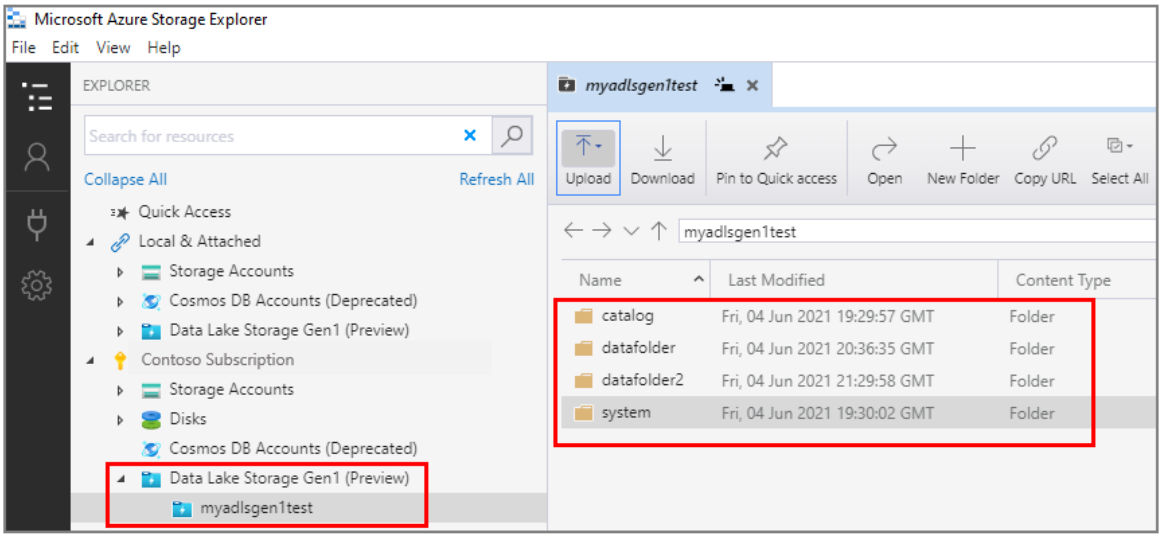


View the contents of a Data Lake Storage Gen1 account

A Data Lake Storage Gen1 account's resources contain folders and files. The following steps show how to view the contents of a Data Lake Storage Gen1 account within Storage Explorer.

1. Open Storage Explorer.
2. Expand the subscription that contains the Data Lake Storage Gen1 account that you want to view.
3. Expand **Data Lake Storage Gen1 (Preview)**.
4. Select the Data Lake Storage Gen1 account that you want to view.

   The main pane displays the contents of the Data Lake Storage Gen1 account.



Manage resources in Data Lake Storage Gen1

You can manage Data Lake Storage Gen1 resources by doing following operations:

- Browse through Data Lake Storage Gen1 resources across multiple Data Lake Storage Gen1 accounts.
- Use a connection string to connect to and manage Data Lake Storage Gen1 directly.
- View Data Lake Storage Gen1 resources shared by others through an ACL under **Local & Attached**.
- Perform file and folder CRUD operations: support recursive folders and multi-selected files.
- Drag, drop, and add a folder to quickly access recent locations. This operation mirrors the desktop File Explorer experience.
- Copy and open a Data Lake Storage Gen1 hyperlink in Storage Explorer with one click.
- Display the **Activities** log in the lower pane to view activity status.

- Display folder statistics and file properties.

Manage resources in Azure Storage Explorer

After you create a Data Lake Storage Gen1 account, you can:

- Upload folders and files, download folders and files, and open resources on your local computer.
- Pin to **Quick Access**, create a new folder, copy a URL, and select all.
- Copy and paste, rename, delete, get folder statistics, and refresh.

The following items show how to manage resources in a Data Lake Storage Gen1 account. Follow the steps for the task that you want to do.

Upload files

1. On the main pane's toolbar, select **Upload**, and then select **Upload Files**.
2. In the **Select files to upload** dialog box, select the files that you want to upload.
3. Select **Open** to begin the upload.

> **NOTE**
> You can also directly drag the files on a local computer to start uploading.

Upload a folder

1. On the main pane's toolbar, select **Upload**, and then select **Upload Folder**.
2. In the **Select folder to upload** dialog box, select the folder that you want to upload.
3. Select **Select Folder** to begin the upload.

> **NOTE**
> You can also directly drag a folder on a local computer to start uploading.

Download folders or files to your local computer

1. Select the folders or files that you want to download.
2. On the main pane's toolbar, select **Download**.
3. In the **Select a folder to save the downloaded files into** dialog box, specify the location and the name.
4. Select **Save**.

Open a folder or file from your local computer

1. Select the folder or file that you want to open.
2. On the main pane's toolbar, select **Open**. Or, right-click the selected folder or file, and then select **Open** on the shortcut menu.

The file is downloaded and opened through the application that's associated with the underlying file type. Or, the folder is opened in the main pane.

Copy folders or files to the clipboard

You can copy Data Lake Storage Gen1 folders or files and paste them in another Data Lake Storage Gen1 account. Copy and paste operations across storage types aren't supported. For example, you can't copy Data Lake Storage Gen1 folders or files and paste them to Azure Blob storage or the other way around.

1. Select the folders or files that you want to copy.
2. On the main pane's toolbar, select **Copy**. Or, right-click the selected folders or files, and then select **Copy** on

the shortcut menu.

3. In the navigation pane, browse to another Data Lake Storage Gen1 account, and select it to view it in the main pane.
4. On the main pane's toolbar, select **Paste** to create a copy. Or, select **Paste** on the destination's shortcut menu.

> **NOTE**
>
> The copy/paste operation works by downloading the folders or files to the local computer and then uploading them to the destination. The tool doesn't perform the action in the back end. The copy/paste operation on large files is slow.

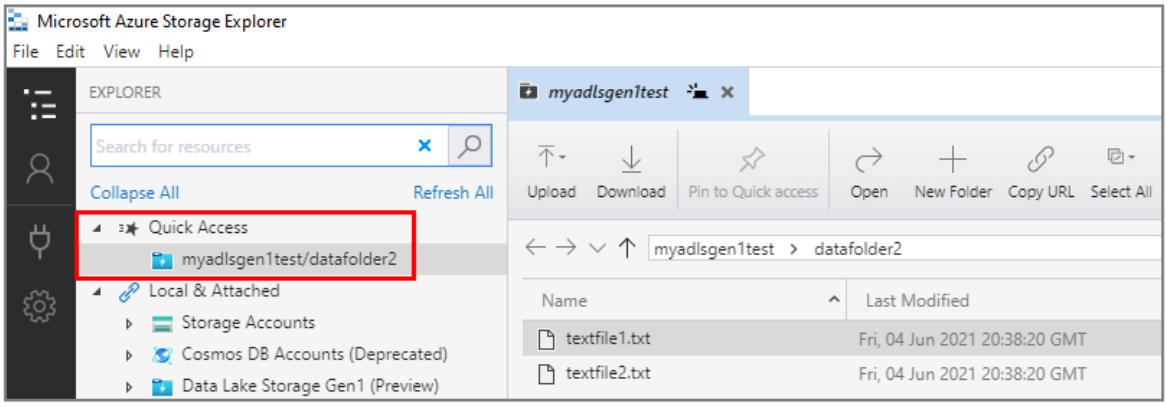
Delete folders or files

1. Select the folders or files that you want to delete.
2. On the main pane's toolbar, select **Delete**. Or, right-click the selected folders or files, and then select **Delete** on the shortcut menu.
3. Select **Yes** in the confirmation dialog box.

Pin to Quick Access

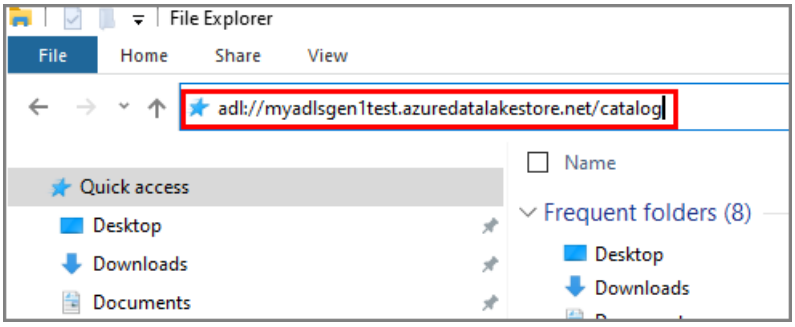
1. Select the folder that you want to pin so that you can easily access the resources.
2. On the main pane's toolbar, select **Pin to Quick access**.

   In the navigation pane, the selected folder is added to the **Quick Access** node.



Use deep links

If you have a URL, you can enter the URL into the address path in File Explorer or a browser. Then Storage Explorer opens automatically and goes to the location of the URL.



Next steps

- View the latest Storage Explorer release notes and videos.
- Get started with Storage Explorer
- Get started with Azure Data Lake Storage Gen1

Copy data from Azure Storage Blobs to Azure Data Lake Storage Gen1

10/3/2022 • 8 minutes to read • Edit Online

Data Lake Storage Gen1 provides a command-line tool, AdlCopy, to copy data from the following sources:

- From Azure Storage blobs into Data Lake Storage Gen1. You can't use AdlCopy to copy data from Data Lake Storage Gen1 to Azure Storage blobs.
- Between two Data Lake Storage Gen1 accounts.

Also, you can use the AdlCopy tool in two different modes:

- **Standalone**, where the tool uses Data Lake Storage Gen1 resources to perform the task.
- **Using a Data Lake Analytics account**, where the units assigned to your Data Lake Analytics account are used to perform the copy operation. You might want to use this option when you are looking to perform the copy tasks in a predictable manner.

Prerequisites

Before you begin this article, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Azure Storage blobs** container with some data.
- **A Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1
- **Data Lake Analytics account (optional)** - See Get started with Azure Data Lake Analytics for instructions on how to create a Data Lake Analytics account.
- **AdlCopy tool**. Install the AdlCopy tool.

Syntax of the AdlCopy tool

Use the following syntax to work with the AdlCopy tool

```
AdlCopy /Source <Blob or Data Lake Storage Gen1 source> /Dest <Data Lake Storage Gen1 destination>
/SourceKey <Key for Blob account> /Account <Data Lake Analytics account> /Units <Number of Analytics units>
/Pattern
```

The parameters in the syntax are described below:

OPTION	DESCRIPTION
Source	Specifies the location of the source data in the Azure storage blob. The source can be a blob container, a blob, or another Data Lake Storage Gen1 account.
Dest	Specifies the Data Lake Storage Gen1 destination to copy to.
SourceKey	Specifies the storage access key for the Azure storage blob source. This is required only if the source is a blob container or a blob.

OPTION	DESCRIPTION
Account	**Optional**. Use this if you want to use Azure Data Lake Analytics account to run the copy job. If you use the /Account option in the syntax but do not specify a Data Lake Analytics account, AdlCopy uses a default account to run the job. Also, if you use this option, you must add the source (Azure Storage Blob) and destination (Azure Data Lake Storage Gen1) as data sources for your Data Lake Analytics account.
Units	Specifies the number of Data Lake Analytics units that will be used for the copy job. This option is mandatory if you use the **/Account** option to specify the Data Lake Analytics account.
Pattern	Specifies a regex pattern that indicates which blobs or files to copy. AdlCopy uses case-sensitive matching. The default pattern when no pattern is specified is to copy all items. Specifying multiple file patterns is not supported.

Use AdlCopy (as standalone) to copy data from an Azure Storage blob

1. Open a command prompt and navigate to the directory where AdlCopy is installed, typically `%HOMEPATH%\Documents\adlcopy`.
2. Run the following command to copy a specific blob from the source container to a Data Lake Storage Gen1 folder:

   ```
   AdlCopy /source https://<source_account>.blob.core.windows.net/<source_container>/<blob name> /dest
   swebhdfs://<dest_adlsg1_account>.azuredatalakestore.net/<dest_folder>/ /sourcekey
   <storage_account_key_for_storage_container>
   ```

   For example:

   ```
   AdlCopy /source
   https://mystorage.blob.core.windows.net/mycluster/HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLo
   g/909f2b.log /dest swebhdfs://mydatalakestorage.azuredatalakestore.net/mynewfolder/ /sourcekey
   uJUfvD6cEvhfLoBae2yyQf8t9/BpbWZ4XoYj4kAS5Jf40pZaMNf0q6a8yqTxktwVgRED4vPHeh/50iS9atS5LQ==
   ```

   > **NOTE**
   > The syntax above specifies the file to be copied to a folder in the Data Lake Storage Gen1 account. AdlCopy tool creates a folder if the specified folder name does not exist.

   You will be prompted to enter the credentials for the Azure subscription under which you have your Data Lake Storage Gen1 account. You will see an output similar to the following:

   ```
   Initializing Copy.
   Copy Started.
   100% data copied.
   Finishing Copy.
   Copy Completed. 1 file copied.
   ```

3. You can also copy all the blobs from one container to the Data Lake Storage Gen1 account using the

following command:

```
AdlCopy /source https://<source_account>.blob.core.windows.net/<source_container>/ /dest
swebhdfs://<dest_adlsg1_account>.azuredatalakestore.net/<dest_folder>/ /sourcekey
<storage_account_key_for_storage_container>
```

For example:

```
AdlCopy /Source https://mystorage.blob.core.windows.net/mycluster/example/data/gutenberg/ /dest
adl://mydatalakestorage.azuredatalakestore.net/mynewfolder/ /sourcekey
uJUfvD6cEvhfLoBae2yyQf8t9/BpbWZ4XoYj4kAS5Jf40pZaMNf0q6a8yqTxktwVgRED4vPHeh/50iS9atS5LQ==
```

Performance considerations

If you are copying from an Azure Blob Storage account, you may be throttled during copy on the blob storage side. This will degrade the performance of your copy job. To learn more about the limits of Azure Blob Storage, see Azure Storage limits at Azure subscription and service limits.

Use AdlCopy (as standalone) to copy data from another Data Lake Storage Gen1 account

You can also use AdlCopy to copy data between two Data Lake Storage Gen1 accounts.

1. Open a command prompt and navigate to the directory where AdlCopy is installed, typically `%HOMEPATH%\Documents\adlcopy`.
2. Run the following command to copy a specific file from one Data Lake Storage Gen1 account to another.

   ```
   AdlCopy /Source adl://<source_adlsg1_account>.azuredatalakestore.net/<path_to_file> /dest
   adl://<dest_adlsg1_account>.azuredatalakestore.net/<path>/
   ```

   For example:

   ```
   AdlCopy /Source adl://mydatastorage.azuredatalakestore.net/mynewfolder/909f2b.log /dest
   adl://mynewdatalakestorage.azuredatalakestore.net/mynewfolder/
   ```

   > **NOTE**
   > The syntax above specifies the file to be copied to a folder in the destination Data Lake Storage Gen1 account. AdlCopy tool creates a folder if the specified folder name does not exist.

   You will be prompted to enter the credentials for the Azure subscription under which you have your Data Lake Storage Gen1 account. You will see an output similar to the following:

   ```
   Initializing Copy.
   Copy Started.|
   100% data copied.
   Finishing Copy.
   Copy Completed. 1 file copied.
   ```

3. The following command copies all files from a specific folder in the source Data Lake Storage Gen1 account to a folder in the destination Data Lake Storage Gen1 account.

```
AdlCopy /Source adl://mydatastorage.azuredatalakestore.net/mynewfolder/ /dest
adl://mynewdatalakestorage.azuredatalakestore.net/mynewfolder/
```

Performance considerations

When using AdlCopy as a standalone tool, the copy is run on shared, Azure-managed resources. The performance you may get in this environment depends on system load and available resources. This mode is best used for small transfers on an ad hoc basis. No parameters need to be tuned when using AdlCopy as a standalone tool.

Use AdlCopy (with Data Lake Analytics account) to copy data

You can also use your Data Lake Analytics account to run the AdlCopy job to copy data from Azure storage blobs to Data Lake Storage Gen1. You would typically use this option when the data to be moved is in the range of gigabytes and terabytes, and you want better and predictable performance throughput.

To use your Data Lake Analytics account with AdlCopy to copy from an Azure Storage Blob, the source (Azure Storage Blob) must be added as a data source for your Data Lake Analytics account. For instructions on adding additional data sources to your Data Lake Analytics account, see Manage Data Lake Analytics account data sources.

> **NOTE**
> If you are copying from an Azure Data Lake Storage Gen1 account as the source using a Data Lake Analytics account, you do not need to associate the Data Lake Storage Gen1 account with the Data Lake Analytics account. The requirement to associate the source store with the Data Lake Analytics account is only when the source is an Azure Storage account.

Run the following command to copy from an Azure Storage blob to a Data Lake Storage Gen1 account using Data Lake Analytics account:

```
AdlCopy /source https://<source_account>.blob.core.windows.net/<source_container>/<blob name> /dest
swebhdfs://<dest_adlsg1_account>.azuredatalakestore.net/<dest_folder>/ /sourcekey
<storage_account_key_for_storage_container> /Account <data_lake_analytics_account> /Units
<number_of_data_lake_analytics_units_to_be_used>
```

For example:

```
AdlCopy /Source https://mystorage.blob.core.windows.net/mycluster/example/data/gutenberg/ /dest
swebhdfs://mydatalakestorage.azuredatalakestore.net/mynewfolder/ /sourcekey
uJUfvD6cEvhfLoBae2yyQf8t9/BpbWZ4XoYj4kAS5Jf40pZaMNf0q6a8yqTxktwVgRED4vPHeh/50iS9atS5LQ== /Account
mydatalakeanalyticaccount /Units 2
```

Similarly, run the following command to copy all files from a specific folder in the source Data Lake Storage Gen1 account to a folder in the destination Data Lake Storage Gen1 account using Data Lake Analytics account:

```
AdlCopy /Source adl://mysourcedatalakestorage.azuredatalakestore.net/mynewfolder/ /dest
adl://mydestdatastorage.azuredatalakestore.net/mynewfolder/ /Account mydatalakeanalyticaccount /Units 2
```

Performance considerations

When copying data in the range of terabytes, using AdlCopy with your own Azure Data Lake Analytics account provides better and more predictable performance. The parameter that should be tuned is the number of Azure Data Lake Analytics Units to use for the copy job. Increasing the number of units will increase the performance of your copy job. Each file to be copied can use maximum one unit. Specifying more units than the number of

files being copied will not increase performance.

Use AdlCopy to copy data using pattern matching

In this section, you learn how to use AdlCopy to copy data from a source (in our example below we use Azure Storage Blob) to a destination Data Lake Storage Gen1 account using pattern matching. For example, you can use the steps below to copy all files with .csv extension from the source blob to the destination.

1. Open a command prompt and navigate to the directory where AdlCopy is installed, typically `%HOMEPATH%\Documents\adlcopy`.
2. Run the following command to copy all files with *.csv extension from a specific blob from the source container to a Data Lake Storage Gen1 folder:

   ```
   AdlCopy /source https://<source_account>.blob.core.windows.net/<source_container>/<blob name> /dest
   swebhdfs://<dest_adlsg1_account>.azuredatalakestore.net/<dest_folder>/ /sourcekey
   <storage_account_key_for_storage_container> /Pattern *.csv
   ```

   For example:

   ```
   AdlCopy /source
   https://mystorage.blob.core.windows.net/mycluster/HdiSamples/HdiSamples/FoodInspectionData/ /dest
   adl://mydatalakestorage.azuredatalakestore.net/mynewfolder/ /sourcekey
   uJUfvD6cEvhfLoBae2yyQf8t9/BpbWZ4XoYj4kAS5Jf40pZaMNf0q6a8yqTxktwVgRED4vPHeh/50iS9atS5LQ== /Pattern
   *.csv
   ```

Billing

- If you use the AdlCopy tool as standalone you will be billed for egress costs for moving data, if the source Azure Storage account is not in the same region as the Data Lake Storage Gen1 account.
- If you use the AdlCopy tool with your Data Lake Analytics account, standard Data Lake Analytics billing rates will apply.

Considerations for using AdlCopy

- AdlCopy (for version 1.0.5), supports copying data from sources that collectively have more than thousands of files and folders. However, if you encounter issues copying a large dataset, you can distribute the files/folders into different subfolders and use the path to those subfolders as the source instead.

Performance considerations for using AdlCopy

AdlCopy supports copying data containing thousands of files and folders. However, if you encounter issues copying a large dataset, you can distribute the files/folders into smaller subfolders. AdlCopy was built for ad hoc copies. If you are trying to copy data on a recurring basis, you should consider using Azure Data Factory that provides full management around the copy operations.

Release notes

- 1.0.13 - If you are copying data to the same Azure Data Lake Storage Gen1 account across multiple adlcopy commands, you do not need to reenter your credentials for each run anymore. Adlcopy will now cache that information across multiple runs.

Next steps

- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Use DistCp to copy data between Azure Storage blobs and Azure Data Lake Storage Gen1

10/3/2022 • 4 minutes to read • Edit Online

If you have an HDInsight cluster with access to Azure Data Lake Storage Gen1, you can use Hadoop ecosystem tools like DistCp to copy data to and from an HDInsight cluster storage (WASB) into a Data Lake Storage Gen1 account. This article shows how to use the DistCp tool.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1.
- **Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. Make sure you enable Remote Desktop for the cluster.

Use DistCp from an HDInsight Linux cluster

An HDInsight cluster comes with the DistCp tool, which can be used to copy data from different sources into an HDInsight cluster. If you've configured the HDInsight cluster to use Data Lake Storage Gen1 as additional storage, you can use DistCp out-of-the-box to copy data to and from a Data Lake Storage Gen1 account. In this section, we look at how to use the DistCp tool.

1. From your desktop, use SSH to connect to the cluster. See Connect to a Linux-based HDInsight cluster. Run the commands from the SSH prompt.
2. Verify whether you can access the Azure Storage blobs (WASB). Run the following command:

   ```
   hdfs dfs -ls wasb://<container_name>@<storage_account_name>.blob.core.windows.net/
   ```

   The output provides a list of contents in the storage blob.
3. Similarly, verify whether you can access the Data Lake Storage Gen1 account from the cluster. Run the following command:

   ```
   hdfs dfs -ls adl://<data_lake_storage_gen1_account>.azuredatalakestore.net:443/
   ```

   The output provides a list of files and folders in the Data Lake Storage Gen1 account.
4. Use DistCp to copy data from WASB to a Data Lake Storage Gen1 account.

   ```
   hadoop distcp
   wasb://<container_name>@<storage_account_name>.blob.core.windows.net/example/data/gutenberg
   adl://<data_lake_storage_gen1_account>.azuredatalakestore.net:443/myfolder
   ```

   The command copies the contents of the **/example/data/gutenberg/** folder in WASB to **/myfolder** in the Data Lake Storage Gen1 account.
5. Similarly, use DistCp to copy data from a Data Lake Storage Gen1 account to WASB.

```
hadoop distcp adl://<data_lake_storage_gen1_account>.azuredatalakestore.net:443/myfolder
wasb://<container_name>@<storage_account_name>.blob.core.windows.net/example/data/gutenberg
```

The command copies the contents of **/myfolder** in the Data Lake Storage Gen1 account to **/example/data/gutenberg/** folder in WASB.

Performance considerations while using DistCp

Because the DistCp tool's lowest granularity is a single file, setting the maximum number of simultaneous copies is the most important parameter to optimize it against Data Lake Storage Gen1. You can control the number of simultaneous copies by setting the number of mappers ('m') parameter on the command line. This parameter specifies the maximum number of mappers that are used to copy data. The default value is 20.

Example:

```
 hadoop distcp wasb://<container_name>@<storage_account_name>.blob.core.windows.net/example/data/gutenberg
adl://<data_lake_storage_gen1_account>.azuredatalakestore.net:443/myfolder -m 100
```

How to determine the number of mappers to use

Here's some guidance that you can use.

- **Step 1: Determine total YARN memory** - The first step is to determine the YARN memory available to the cluster where you run the DistCp job. This information is available in the Ambari portal associated with the cluster. Navigate to YARN and view the **Configs** tab to see the YARN memory. To get the total YARN memory, multiply the YARN memory per node with the number of nodes you have in your cluster.
- **Step 2: Calculate the number of mappers** - The value of **m** is equal to the quotient of total YARN memory divided by the YARN container size. The YARN container size information is also available in the Ambari portal. Navigate to YARN and view the **Configs** tab. The YARN container size is displayed in this window. The equation to arrive at the number of mappers (**m**) is:

  ```
  m = (number of nodes * YARN memory for each node) / YARN container size
  ```

Example:

Let's assume that you have four D14v2s nodes in the cluster and you want to transfer 10 TB of data from 10 different folders. Each of the folders contains varying amounts of data and the file sizes within each folder are different.

- Total YARN memory - From the Ambari portal you determine that the YARN memory is 96 GB for a D14 node. So, total YARN memory for four node cluster is:

  ```
  YARN memory = 4 * 96GB = 384GB
  ```

- Number of mappers - From the Ambari portal you determine that the YARN container size is 3072 for a D14 cluster node. So, the number of mappers is:

  ```
  m = (4 nodes * 96GB) / 3072MB = 128 mappers
  ```

If other applications are using memory, you can choose to only use a portion of your cluster's YARN memory for DistCp.

Copying large datasets

When the size of the dataset to be moved is large (for example, > 1 TB) or if you have many different folders, consider using multiple DistCp jobs. There's likely no performance gain, but it spreads out the jobs so that if any job fails, you need to only restart that specific job instead of the entire job.

Limitations

- DistCp tries to create mappers that are similar in size to optimize performance. Increasing the number of mappers may not always increase performance.
- DistCp is limited to only one mapper per file. Therefore, you shouldn't have more mappers than you have files. Because DistCp can assign only one mapper to a file, this limits the amount of concurrency that can be used to copy large files.
- If you have a small number of large files, split them into 256-MB file chunks to give you more potential concurrency.
- If you're copying from an Azure Blob storage account, your copy job may be throttled on the Blob storage side. This degrades the performance of your copy job. To learn more about the limits of Azure Blob storage, see Azure Storage limits at Azure subscription and service limits.

See also

- Copy data from Azure Storage blobs to Data Lake Storage Gen1
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Copy data between Data Lake Storage Gen1 and Azure SQL Database using Sqoop

10/3/2022 • 4 minutes to read • Edit Online

Learn how to use Apache Sqoop to import and export data between Azure SQL Database and Azure Data Lake Storage Gen1.

What is Sqoop?

Big data applications are a natural choice for processing unstructured and semi-structured data, such as logs and files. However, you may also have a need to process structured data that's stored in relational databases.

Apache Sqoop is a tool designed to transfer data between relational databases and a big data repository, such as Data Lake Storage Gen1. You can use it to import data from a relational database management system (RDBMS) such as Azure SQL Database into Data Lake Storage Gen1. You can then transform and analyze the data using big data workloads, and then export the data back into an RDBMS. In this article, you use a database in Azure SQL Database as your relational database to import/export from.

Prerequisites

Before you begin, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create the account, see Get started with Azure Data Lake Storage Gen1
- **Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. This article assumes you have an HDInsight Linux cluster with Data Lake Storage Gen1 access.
- **Azure SQL Database**. For instructions on how to create a database in Azure SQL Database, see Create a database in Azure SQL Database

Create sample tables in the database

1. To start, create two sample tables in the database. Use SQL Server Management Studio or Visual Studio to connect to the database and then run the following queries.

   Create Table1

   ```
   CREATE TABLE [dbo].[Table1](
   [ID] [int] NOT NULL,
   [FName] [nvarchar](50) NOT NULL,
   [LName] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_Table_1] PRIMARY KEY CLUSTERED
          (
                [ID] ASC
          )
   ) ON [PRIMARY]
   GO
   ```

   Create Table2

```
CREATE TABLE [dbo].[Table2](
[ID] [int] NOT NULL,
[FName] [nvarchar](50) NOT NULL,
[LName] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Table_2] PRIMARY KEY CLUSTERED
       (
              [ID] ASC
       )
) ON [PRIMARY]
GO
```

2. Run the following command to add some sample data to **Table1**. Leave **Table2** empty. Later, you'll import data from **Table1** into Data Lake Storage Gen1. Then, you'll export data from Data Lake Storage Gen1 into **Table2**.

```
INSERT INTO [dbo].[Table1] VALUES (1,'Neal','Kell'), (2,'Lila','Fulton'), (3, 'Erna','Myers'),
(4,'Annette','Simpson');
```

Use Sqoop from an HDInsight cluster with access to Data Lake Storage Gen1

An HDInsight cluster already has the Sqoop packages available. If you've configured the HDInsight cluster to use Data Lake Storage Gen1 as additional storage, you can use Sqoop (without any configuration changes) to import/export data between a relational database such as Azure SQL Database, and a Data Lake Storage Gen1 account.

1. For this article, we assume you created a Linux cluster so you should use SSH to connect to the cluster. See Connect to a Linux-based HDInsight cluster.

2. Verify whether you can access the Data Lake Storage Gen1 account from the cluster. Run the following command from the SSH prompt:

```
hdfs dfs -ls adl://<data_lake_storage_gen1_account>.azuredatalakestore.net/
```

   This command provides a list of files/folders in the Data Lake Storage Gen1 account.

Import data from Azure SQL Database into Data Lake Storage Gen1

1. Navigate to the directory where Sqoop packages are available. Typically, this location is `/usr/hdp/<version>/sqoop/bin`.

2. Import the data from **Table1** into the Data Lake Storage Gen1 account. Use the following syntax:

```
sqoop-import --connect "jdbc:sqlserver://<sql-database-server-
name>.database.windows.net:1433;username=<username>@<sql-database-server-name>;password=
<password>;database=<sql-database-name>" --table Table1 --target-dir adl://<data-lake-storage-gen1-
name>.azuredatalakestore.net/Sqoop/SqoopImportTable1
```

   The **sql-database-server-name** placeholder represents the name of the server where the database is running. **sql-database-name** placeholder represents the actual database name.

   For example,

```
sqoop-import --connect
"jdbc:sqlserver://mysqoopserver.database.windows.net:1433;username=user1@mysqoopserver;password=
<password>;database=mysqoopdatabase" --table Table1 --target-dir
adl://myadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1
```

3. Verify that the data has been transferred to the Data Lake Storage Gen1 account. Run the following command:

```
hdfs dfs -ls adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/
```

You should see the following output.

```
-rwxrwxrwx   0 sshuser hdfs          0 2016-02-26 21:09
adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/_SUCCESS
-rwxrwxrwx   0 sshuser hdfs         12 2016-02-26 21:09
adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/part-m-00000
-rwxrwxrwx   0 sshuser hdfs         14 2016-02-26 21:09
adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/part-m-00001
-rwxrwxrwx   0 sshuser hdfs         13 2016-02-26 21:09
adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/part-m-00002
-rwxrwxrwx   0 sshuser hdfs         18 2016-02-26 21:09
adl://hdiadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1/part-m-00003
```

Each **part-m-*** file corresponds to a row in the source table, **Table1**. You can view the contents of the part-m-* files to verify.

Export data from Data Lake Storage Gen1 into Azure SQL Database

1. Export the data from the Data Lake Storage Gen1 account to the empty table, **Table2**, in the Azure SQL Database. Use the following syntax.

```
sqoop-export --connect "jdbc:sqlserver://<sql-database-server-
name>.database.windows.net:1433;username=<username>@<sql-database-server-name>;password=
<password>;database=<sql-database-name>" --table Table2 --export-dir adl://<data-lake-storage-gen1-
name>.azuredatalakestore.net/Sqoop/SqoopImportTable1 --input-fields-terminated-by ","
```

For example,

```
sqoop-export --connect
"jdbc:sqlserver://mysqoopserver.database.windows.net:1433;username=user1@mysqoopserver;password=
<password>;database=mysqoopdatabase" --table Table2 --export-dir
adl://myadlsg1store.azuredatalakestore.net/Sqoop/SqoopImportTable1 --input-fields-terminated-by ","
```

2. Verify that the data was uploaded to the SQL Database table. Use SQL Server Management Studio or Visual Studio to connect to the Azure SQL Database and then run the following query.

```
SELECT * FROM TABLE2
```

This command should have the following output.

```
 ID  FName    LName
------------------
1    Neal     Kell
2    Lila     Fulton
3    Erna     Myers
4    Annette  Simpson
```

Performance considerations while using Sqoop

For information about performance tuning your Sqoop job to copy data to Data Lake Storage Gen1, see the Sqoop performance blog post.

Next steps

- Copy data from Azure Storage Blobs to Data Lake Storage Gen1
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Use the Azure Import/Export service for offline copy of data to Data Lake Storage Gen1

10/3/2022 • 4 minutes to read • Edit Online

In this article, you'll learn how to copy huge data sets (>200 GB) into Data Lake Storage Gen1 by using offline copy methods, like the Azure Import/Export service. Specifically, the file used as an example in this article is 339,420,860,416 bytes, or about 319 GB on disk. Let's call this file 319GB.tsv.

The Azure Import/Export service helps you to transfer large amounts of data more securely to Azure Blob storage by shipping hard disk drives to an Azure datacenter.

Prerequisites

Before you begin, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **An Azure storage account**.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1.

Prepare the data

Before using the Import/Export service, break the data file to be transferred **into copies that are less than 200 GB** in size. The import tool does not work with files greater than 200 GB. In this article, we split the file into chunks of 100 GB each. You can do this by using Cygwin. Cygwin supports Linux commands. In this case, use the following command:

```
split -b 100m 319GB.tsv
```

The split operation creates files with the following names.

- *319GB.tsv-part-aa*
- *319GB.tsv-part-ab*
- *319GB.tsv-part-ac*
- *319GB.tsv-part-ad*

Get disks ready with data

Follow the instructions in Using the Azure Import/Export service (under the **Prepare your drives** section) to prepare your hard drives. Here's the overall sequence:

1. Procure a hard disk that meets the requirement to be used for the Azure Import/Export service.
2. Identify an Azure storage account where the data will be copied after it is shipped to the Azure datacenter.
3. Use the Azure Import/Export Tool, a command-line utility. Here's a sample snippet that shows how to use the tool.

```
WAImportExport PrepImport /sk:<StorageAccountKey> /t: <TargetDriveLetter> /format /encrypt
/logdir:e:\myexportimportjob\logdir /j:e:\myexportimportjob\journal1.jrn /id:myexportimportjob
/srcdir:F:\demo\ExImContainer /dstdir:importcontainer/vf1/
```

See Using the Azure Import/Export service for more sample snippets.

4. The preceding command creates a journal file at the specified location. Use this journal file to create an import job from the Azure portal.

Create an import job

You can now create an import job by using the instructions in Using the Azure Import/Export service (under the **Create the Import job** section). For this import job, with other details, also provide the journal file created while preparing the disk drives.

Physically ship the disks

You can now physically ship the disks to an Azure datacenter. There, the data is copied over to the Azure Storage blobs you provided while creating the import job. Also, while creating the job, if you opted to provide the tracking information later, you can now go back to your import job and update the tracking number.

Copy data from blobs to Data Lake Storage Gen1

After the status of the import job shows that it's completed, you can verify whether the data is available in the Azure Storage blobs you had specified. You can then use a variety of methods to move that data from the blobs to Azure Data Lake Storage Gen1. For all the available options for uploading data, see Ingesting data into Data Lake Storage Gen1.

In this section, we provide you with the JSON definitions that you can use to create an Azure Data Factory pipeline for copying data. You can use these JSON definitions from the Azure portal or Visual Studio.

Source linked service (Azure Storage blob)

```
{
    "name": "AzureStorageLinkedService",
    "properties": {
        "type": "AzureStorage",
        "description": "",
        "typeProperties": {
            "connectionString": "DefaultEndpointsProtocol=https;AccountName=<accountname>;AccountKey=
<accountkey>"
        }
    }
}
```

Target linked service (Data Lake Storage Gen1)

```
{
    "name": "AzureDataLakeStorageGen1LinkedService",
    "properties": {
        "type": "AzureDataLakeStore",
        "description": "",
        "typeProperties": {
            "authorization": "<Click 'Authorize' to allow this data factory and the activities it runs to
access this Data Lake Storage Gen1 account with your access rights>",
            "dataLakeStoreUri": "https://<adlsg1_account_name>.azuredatalakestore.net/webhdfs/v1",
            "sessionId": "<OAuth session id from the OAuth authorization session. Each session id is unique
and may only be used once>"
        }
    }
}
```

Input data set

```
{
    "name": "InputDataSet",
    "properties": {
        "published": false,
        "type": "AzureBlob",
        "linkedServiceName": "AzureStorageLinkedService",
        "typeProperties": {
            "folderPath": "importcontainer/vf1/"
        },
        "availability": {
            "frequency": "Hour",
            "interval": 1
        },
        "external": true,
        "policy": {}
    }
}
```

Output data set

```
{
"name": "OutputDataSet",
"properties": {
  "published": false,
  "type": "AzureDataLakeStore",
  "linkedServiceName": "AzureDataLakeStorageGen1LinkedService",
  "typeProperties": {
    "folderPath": "/importeddatafeb8job/"
    },
  "availability": {
    "frequency": "Hour",
    "interval": 1
    }
  }
}
```

Pipeline (copy activity)

```
{
    "name": "CopyImportedData",
    "properties": {
        "description": "Pipeline with copy activity",
        "activities": [
            {
                "type": "Copy",
                "typeProperties": {
                    "source": {
                        "type": "BlobSource"
                    },
                    "sink": {
                        "type": "AzureDataLakeStoreSink",
                        "copyBehavior": "PreserveHierarchy",
                        "writeBatchSize": 0,
                        "writeBatchTimeout": "00:00:00"
                    }
                },
                "inputs": [
                    {
                        "name": "InputDataSet"
                    }
                ],
                "outputs": [
                    {
                        "name": "OutputDataSet"
                    }
                ],
                "policy": {
                    "timeout": "01:00:00",
                    "concurrency": 1
                },
                "scheduler": {
                    "frequency": "Hour",
                    "interval": 1
                },
                "name": "AzureBlobtoDataLake",
                "description": "Copy Activity"
            }
        ],
        "start": "2016-02-08T22:00:00Z",
        "end": "2016-02-08T23:00:00Z",
        "isPaused": false,
        "pipelineMode": "Scheduled"
    }
}
```

For more information, see Move data from Azure Storage blob to Azure Data Lake Storage Gen1 using Azure Data Factory.

Reconstruct the data files in Data Lake Storage Gen1

> **NOTE**
>
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

We started with a file that was 319 GB, and broke it down into files of smaller size so that it could be transferred by using the Azure Import/Export service. Now that the data is in Azure Data Lake Storage Gen1, we can reconstruct the file to its original size. You can use the following Azure PowerShell cmdlets to do so.

```
# Login to our account
Connect-AzAccount

# List your subscriptions
Get-AzSubscription

# Switch to the subscription you want to work with
Set-AzContext -SubscriptionId
Register-AzResourceProvider -ProviderNamespace "Microsoft.DataLakeStore"

# Join  the files
Join-AzDataLakeStoreItem -AccountName "<adlsg1_account_name" -Paths "/importeddatafeb8job/319GB.tsv-part-
aa","/importeddatafeb8job/319GB.tsv-part-ab", "/importeddatafeb8job/319GB.tsv-part-ac",
"/importeddatafeb8job/319GB.tsv-part-ad" -Destination "/importeddatafeb8job/MergedFile.csv"
```

Next steps

- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Migrate Azure Data Lake Storage Gen1 across regions

10/3/2022 • 3 minutes to read • Edit Online

As Azure Data Lake Storage Gen1 becomes available in new regions, you might choose to do a one-time migration, to take advantage of the new region. Learn what to consider as you plan and complete the migration.

Prerequisites

- **An Azure subscription**. For more information, see Create your free Azure account today.
- **A Data Lake Storage Gen1 account in two different regions**. For more information, see Get started with Azure Data Lake Storage Gen1.
- **Azure Data Factory**. For more information, see Introduction to Azure Data Factory.

Migration considerations

First, identify the migration strategy that works best for your application that writes, reads, or processes data in Data Lake Storage Gen1. When you choose a strategy, consider your application's availability requirements, and the downtime that occurs during a migration. For example, your simplest approach might be to use the "lift-and-shift" cloud migration model. In this approach, you pause the application in your existing region while all your data is copied to the new region. When the copy process is finished, you resume your application in the new region, and then delete the old Data Lake Storage Gen1 account. Downtime during the migration is required.

To reduce downtime, you might immediately start ingesting new data in the new region. When you have the minimum data needed, run your application in the new region. In the background, continue to copy older data from the existing Data Lake Storage Gen1 account to the new Data Lake Storage Gen1 account in the new region. By using this approach, you can make the switch to the new region with little downtime. When all the older data has been copied, delete the old Data Lake Storage Gen1 account.

Other important details to consider when planning your migration are:

- **Data volume**. The volume of data (in gigabytes, the number of files and folders, and so on) affects the time and resources you need for the migration.
- **Data Lake Storage Gen1 account name**. The new account name in the new region must be globally unique. For example, the name of your old Data Lake Storage Gen1 account in East US 2 might be contosoeastus2.azuredatalakestore.net. You might name your new Data Lake Storage Gen1 account in North EU contosonortheu.azuredatalakestore.net.
- **Tools**. We recommend that you use the Azure Data Factory Copy Activity to copy Data Lake Storage Gen1 files. Data Factory supports data movement with high performance and reliability. Keep in mind that Data Factory copies only the folder hierarchy and content of the files. You need to manually apply any access control lists (ACLs) that you use in the old account to the new account. For more information, including performance targets for best-case scenarios, see the Copy Activity performance and tuning guide. If you want data copied more quickly, you might need to use additional Cloud Data Movement Units. Some other tools, like AdlCopy, don't support copying data between regions.
- **Bandwidth charges**. Bandwidth charges apply because data is transferred out of an Azure region.
- **ACLs on your data**. Secure your data in the new region by applying ACLs to files and folders. For more information, see Securing data stored in Azure Data Lake Storage Gen1. We recommend that you use the

migration to update and adjust your ACLs. You might want to use settings similar to your current settings. You can view the ACLs that are applied to any file by using the Azure portal, PowerShell cmdlets, or SDKs.

- **Location of analytics services**. For best performance, your analytics services, like Azure Data Lake Analytics or Azure HDInsight, should be in the same region as your data.

Next steps

- Overview of Azure Data Lake Storage Gen1

Security in Azure Data Lake Storage Gen1

10/3/2022 • 7 minutes to read • Edit Online

Many enterprises are taking advantage of big data analytics for business insights to help them make smart decisions. An organization might have a complex and regulated environment, with an increasing number of diverse users. It is vital for an enterprise to make sure that critical business data is stored more securely, with the correct level of access granted to individual users. Azure Data Lake Storage Gen1 is designed to help meet these security requirements. In this article, learn about the security capabilities of Data Lake Storage Gen1, including:

- Authentication
- Authorization
- Network isolation
- Data protection
- Auditing

Authentication and identity management

Authentication is the process by which a user's identity is verified when the user interacts with Data Lake Storage Gen1 or with any service that connects to Data Lake Storage Gen1. For identity management and authentication, Data Lake Storage Gen1 uses Azure Active Directory, a comprehensive identity and access management cloud solution that simplifies the management of users and groups.

Each Azure subscription can be associated with an instance of Azure Active Directory. Only users and service identities that are defined in your Azure Active Directory service can access your Data Lake Storage Gen1 account, by using the Azure portal, command-line tools, or through client applications your organization builds by using the Data Lake Storage Gen1 SDK. Key advantages of using Azure Active Directory as a centralized access control mechanism are:

- Simplified identity lifecycle management. The identity of a user or a service (a service principal identity) can be quickly created and quickly revoked by simply deleting or disabling the account in the directory.
- Multi-factor authentication. Multi-factor authentication provides an additional layer of security for user sign-ins and transactions.
- Authentication from any client through a standard open protocol, such as OAuth or OpenID.
- Federation with enterprise directory services and cloud identity providers.

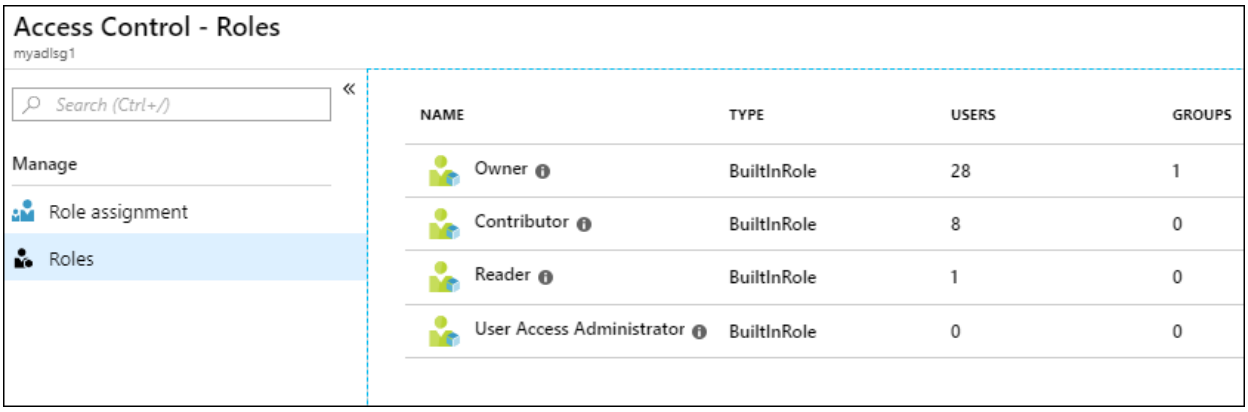
Authorization and access control

After Azure Active Directory authenticates a user so that the user can access Data Lake Storage Gen1, authorization controls access permissions for Data Lake Storage Gen1. Data Lake Storage Gen1 separates authorization for account-related and data-related activities in the following manner:

- Azure role-based access control (Azure RBAC) for account management
- POSIX ACL for accessing data in the store

Azure RBAC for account management

Four basic roles are defined for Data Lake Storage Gen1 by default. The roles permit different operations on a Data Lake Storage Gen1 account via the Azure portal, PowerShell cmdlets, and REST APIs. The Owner and Contributor roles can perform a variety of administration functions on the account. You can assign the Reader role to users who only view account management data.



Note that although roles are assigned for account management, some roles affect access to data. You need to use ACLs to control access to operations that a user can perform on the file system. The following table shows a summary of management rights and data access rights for the default roles.

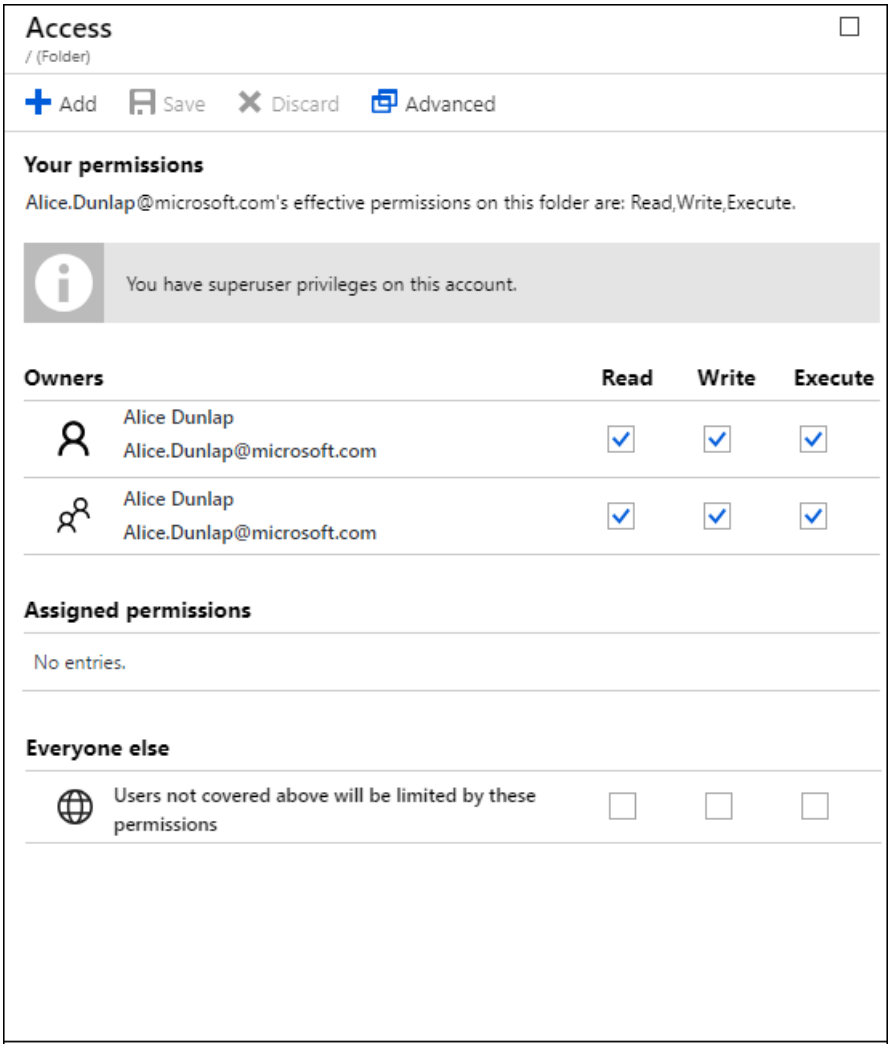
ROLES	MANAGEMENT RIGHTS	DATA ACCESS RIGHTS	EXPLANATION
No role assigned	None	Governed by ACL	The user cannot use the Azure portal or Azure PowerShell cmdlets to browse Data Lake Storage Gen1. The user can use command-line tools only.
Owner	All	All	The Owner role is a superuser. This role can manage everything and has full access to data.
Reader	Read-only	Governed by ACL	The Reader role can view everything regarding account management, such as which user is assigned to which role. The Reader role can't make any changes.
Contributor	All except add and remove roles	Governed by ACL	The Contributor role can manage some aspects of an account, such as deployments and creating and managing alerts. The Contributor role cannot add or remove roles.
User Access Administrator	Add and remove roles	Governed by ACL	The User Access Administrator role can manage user access to accounts.

For instructions, see Assign users or security groups to Data Lake Storage Gen1 accounts.

Using ACLs for operations on file systems

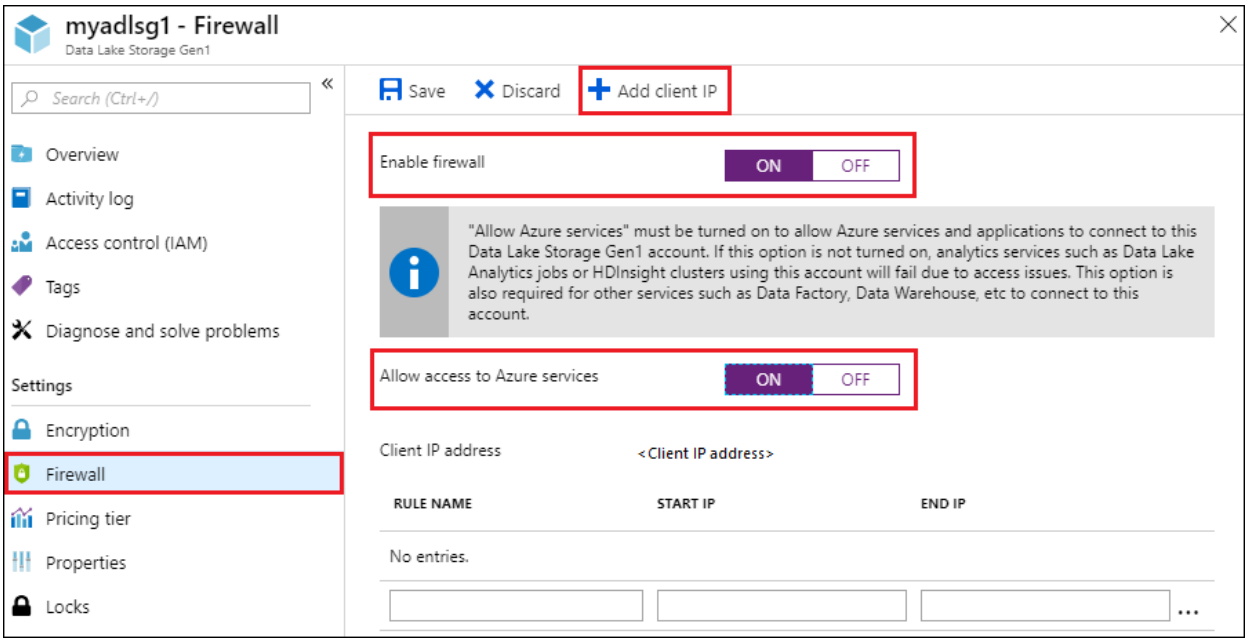
Data Lake Storage Gen1 is a hierarchical file system like Hadoop Distributed File System (HDFS), and it supports POSIX ACLs. It controls read (r), write (w), and execute (x) permissions to resources for the Owner role, for the Owners group, and for other users and groups. In Data Lake Storage Gen1, ACLs can be enabled on the root folder, on subfolders, and on individual files. For more information on how ACLs work in context of Data Lake Storage Gen1, see Access control in Data Lake Storage Gen1.

We recommend that you define ACLs for multiple users by using security groups. Add users to a security group, and then assign the ACLs for a file or folder to that security group. This is useful when you want to provide assigned permissions, because you are limited to a maximum of 28 entries for assigned permissions. For more information about how to better secure data stored in Data Lake Storage Gen1 by using Azure Active Directory security groups, see Assign users or security group as ACLs to the Data Lake Storage Gen1 file system.



Network isolation

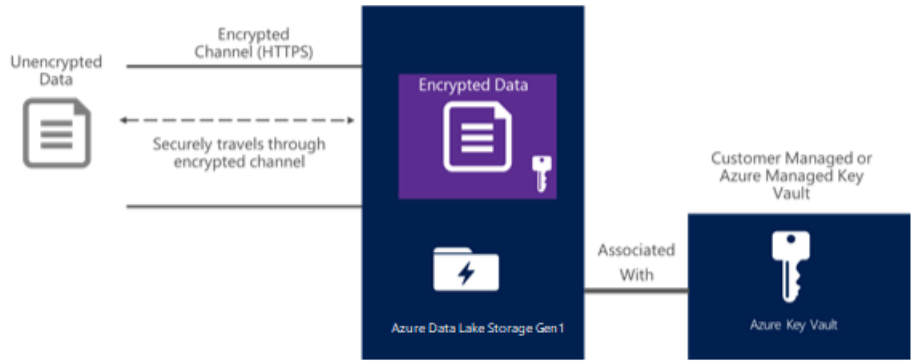
Use Data Lake Storage Gen1 to help control access to your data store at the network level. You can establish firewalls and define an IP address range for your trusted clients. With an IP address range, only clients that have an IP address within the defined range can connect to Data Lake Storage Gen1.



Azure virtual networks (VNet) support service tags for Data Lake Gen 1. A service tag represents a group of IP address prefixes from a given Azure service. Microsoft manages the address prefixes encompassed by the service tag and automatically updates the service tag as addresses change. For more information, see Azure service tags overview.

Data protection

Data Lake Storage Gen1 protects your data throughout its life cycle. For data in transit, Data Lake Storage Gen1 uses the industry-standard Transport Layer Security (TLS 1.2) protocol to secure data over the network.



Data Lake Storage Gen1 also provides encryption for data that is stored in the account. You can chose to have your data encrypted or opt for no encryption. If you opt in for encryption, data stored in Data Lake Storage Gen1 is encrypted prior to storing on persistent media. In such a case, Data Lake Storage Gen1 automatically encrypts data prior to persisting and decrypts data prior to retrieval, so it is completely transparent to the client accessing the data. There is no code change required on the client side to encrypt/decrypt data.

For key management, Data Lake Storage Gen1 provides two modes for managing your master encryption keys (MEKs), which are required for decrypting any data that is stored in Data Lake Storage Gen1. You can either let Data Lake Storage Gen1 manage the MEKs for you, or choose to retain ownership of the MEKs using your Azure Key Vault account. You specify the mode of key management while creating a Data Lake Storage Gen1 account. For more information on how to provide encryption-related configuration, see Get started with Azure Data Lake Storage Gen1 using the Azure Portal.

Activity and diagnostic logs

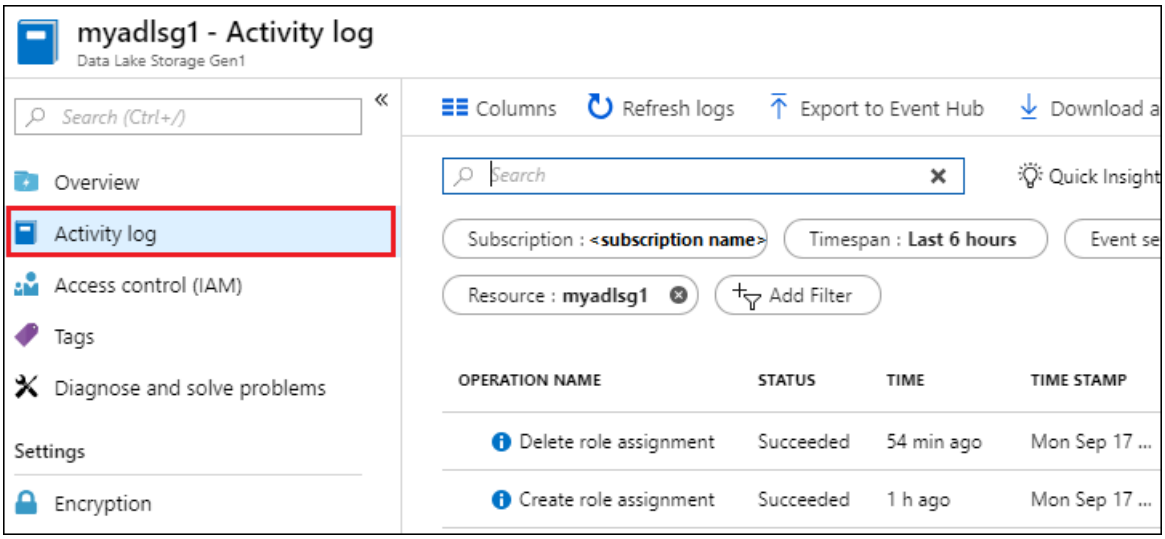
You can use activity or diagnostic logs, depending on whether you are looking for logs for account management-related activities or data-related activities.

- Account management-related activities use Azure Resource Manager APIs and are surfaced in the Azure portal via activity logs.
- Data-related activities use WebHDFS REST APIs and are surfaced in the Azure portal via diagnostic logs.

Activity log

To comply with regulations, an organization might require adequate audit trails of account management activities if it needs to dig into specific incidents. Data Lake Storage Gen1 has built-in monitoring and it logs all account management activities.

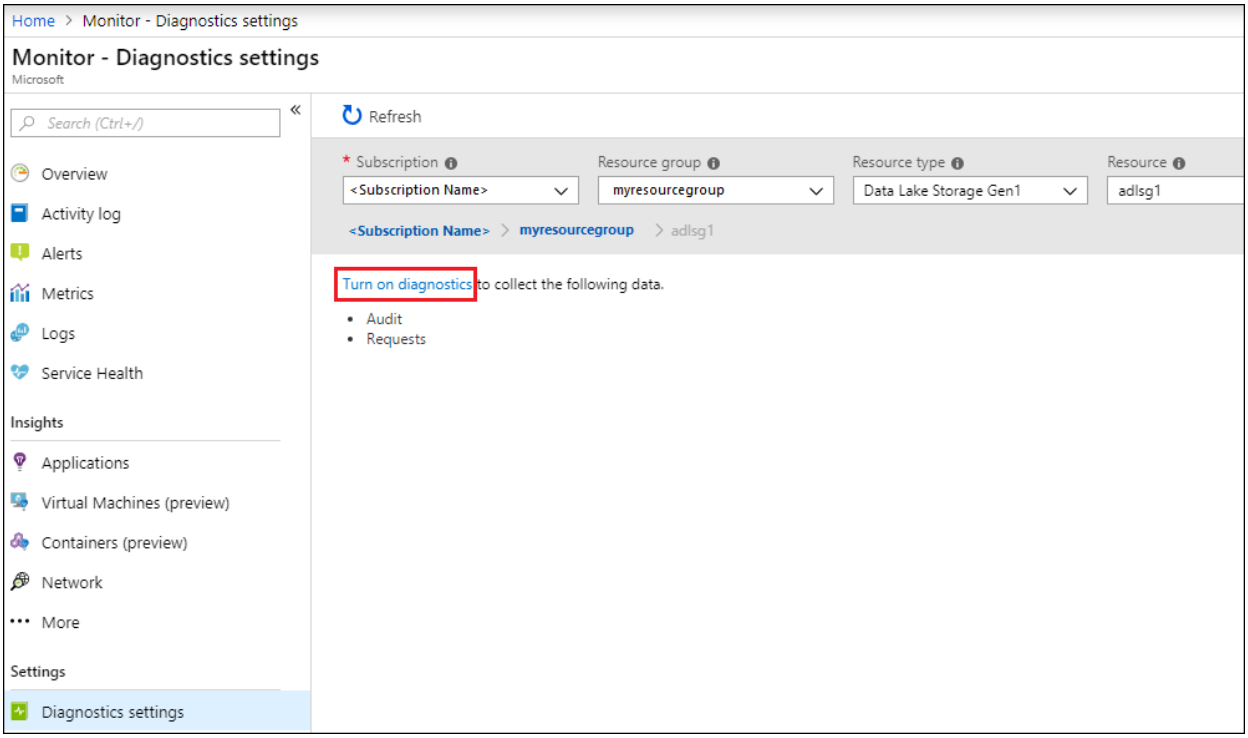
For account management audit trails, view and choose the columns that you want to log. You also can export activity logs to Azure Storage.



For more information on working with activity logs, see View activity logs to audit actions on resources.

Diagnostics logs

You can enable data access audit and diagnostic logging in the Azure portal and send the logs to an Azure Blob storage account, an event hub, or Azure Monitor logs.



For more information on working with diagnostic logs with Data Lake Storage Gen1, see Accessing diagnostic logs for Data Lake Storage Gen1.

Summary

Enterprise customers demand a data analytics cloud platform that is secure and easy to use. Data Lake Storage Gen1 is designed to help address these requirements through identity management and authentication via Azure Active Directory integration, ACL-based authorization, network isolation, data encryption in transit and at rest, and auditing.

If you want to see new features in Data Lake Storage Gen1, send us your feedback in the Data Lake Storage Gen1 UserVoice forum.

See also

- Overview of Azure Data Lake Storage Gen1
- Get started with Data Lake Storage Gen1
- Secure data in Data Lake Storage Gen1

Access control in Azure Data Lake Storage Gen1

10/3/2022 • 10 minutes to read • Edit Online

Azure Data Lake Storage Gen1 implements an access control model that derives from HDFS, which in turn derives from the POSIX access control model. This article summarizes the basics of the access control model for Data Lake Storage Gen1.

Access control lists on files and folders

There are two kinds of access control lists (ACLs), **Access ACLs** and **Default ACLs**.

- **Access ACLs**: These control access to an object. Files and folders both have Access ACLs.
- **Default ACLs**: A "template" of ACLs associated with a folder that determine the Access ACLs for any child items that are created under that folder. Files do not have Default ACLs.

Both Access ACLs and Default ACLs have the same structure.

> **NOTE**
> Changing the Default ACL on a parent does not affect the Access ACL or Default ACL of child items that already exist.

Permissions

The permissions on a filesystem object are **Read**, **Write**, and **Execute**, and they can be used on files and folders as shown in the following table:

	FILE	FOLDER
Read (R)	Can read the contents of a file	Requires **Read** and **Execute** to list the contents of the folder
Write (W)	Can write or append to a file	Requires **Write** and **Execute** to create child items in a folder
Execute (X)	Does not mean anything in the context of Data Lake Storage Gen1	Required to traverse the child items of a folder

Short forms for permissions

RWX is used to indicate **Read + Write + Execute**. A more condensed numeric form exists in which **Read=4**, **Write=2**, and **Execute=1**, the sum of which represents the permissions. Following are some examples.

NUMERIC FORM	SHORT FORM	WHAT IT MEANS
7	`RWX`	Read + Write + Execute
5	`R-X`	Read + Execute
4	`R--`	Read

NUMERIC FORM	SHORT FORM	WHAT IT MEANS
0	`---`	No permissions

Permissions do not inherit

In the POSIX-style model that's used by Data Lake Storage Gen1, permissions for an item are stored on the item itself. In other words, permissions for an item cannot be inherited from the parent items.

Common scenarios related to permissions

Following are some common scenarios to help you understand which permissions are needed to perform certain operations on a Data Lake Storage Gen1 account.

OPERATION	OBJECT	/	SEATTLE/	PORTLAND/	DATA.TXT
Read	Data.txt	`--X`	`--X`	`--X`	`R--`
Append to	Data.txt	`--X`	`--X`	`--X`	`-W-`
Delete	Data.txt	`--X`	`--X`	`-WX`	`---`
Create	Data.txt	`--X`	`--X`	`-WX`	`---`
List	/	`R-X`	`---`	`---`	`---`
List	/Seattle/	`--X`	`R-X`	`---`	`---`
List	/Seattle/Portland/	`--X`	`--X`	`R-X`	`---`

> **NOTE**
> Write permissions on the file are not required to delete it as long as the previous two conditions are true.

Users and identities

Every file and folder has distinct permissions for these identities:

- The owning user
- The owning group
- Named users
- Named groups
- All other users

The identities of users and groups are Azure Active Directory (Azure AD) identities. So unless otherwise noted, a "user," in the context of Data Lake Storage Gen1, can either mean an Azure AD user or an Azure AD security group.

The super-user

A super-user has the most rights of all the users in the Data Lake Storage Gen1 account. A super-user:

- Has RWX Permissions to **all** files and folders.

- Can change the permissions on any file or folder.
- Can change the owning user or owning group of any file or folder.

All users that are part of the **Owners** role for a Data Lake Storage Gen1 account are automatically a super-user.

The owning user

The user who created the item is automatically the owning user of the item. An owning user can:

- Change the permissions of a file that is owned.
- Change the owning group of a file that is owned, as long as the owning user is also a member of the target group.

> **NOTE**
> The owning user *cannot* change the owning user of a file or folder. Only super-users can change the owning user of a file or folder.

The owning group

Background

In the POSIX ACLs, every user is associated with a "primary group." For example, user "alice" might belong to the "finance" group. Alice might also belong to multiple groups, but one group is always designated as her primary group. In POSIX, when Alice creates a file, the owning group of that file is set to her primary group, which in this case is "finance." The owning group otherwise behaves similarly to assigned permissions for other users/groups.

Because there is no “primary group” associated to users in Data Lake Storage Gen1, the owning group is assigned as below.

Assigning the owning group for a new file or folder

- **Case 1**: The root folder "/". This folder is created when a Data Lake Storage Gen1 account is created. In this case, the owning group is set to an all-zero GUID. This value does not permit any access. It is a placeholder until such time a group is assigned.
- **Case 2** (Every other case): When a new item is created, the owning group is copied from the parent folder.

Changing the owning group

The owning group can be changed by:

- Any super-users.
- The owning user, if the owning user is also a member of the target group.

> **NOTE**
> The owning group *cannot* change the ACLs of a file or folder.
>
> For accounts created on or before September 2018, the owning group was set to the user who created the account in the case of the root folder for **Case 1**, above. A single user account is not valid for providing permissions via the owning group, thus no permissions are granted by this default setting. You can assign this permission to a valid user group.

Access check algorithm

The following pseudocode represents the access check algorithm for Data Lake Storage Gen1 accounts.

```
def access_check( user, desired_perms, path ) :
  # access_check returns true if user has the desired permissions on the path, false otherwise
  # user is the identity that wants to perform an operation on path
  # desired_perms is a simple integer with values from 0 to 7 ( R=4, W=2, X=1). User desires these
permissions
  # path is the file or folder
  # Note: the "sticky bit" is not illustrated in this algorithm

# Handle super users.
  if (is_superuser(user)) :
    return True

  # Handle the owning user. Note that mask IS NOT used.
  entry = get_acl_entry( path, OWNER )
  if (user == entry.identity)
      return ( (desired_perms & entry.permissions) == desired_perms )

  # Handle the named users. Note that mask IS used.
  entries = get_acl_entries( path, NAMED_USER )
  for entry in entries:
      if (user == entry.identity ) :
          mask = get_mask( path )
          return ( (desired_perms & entry.permmissions & mask) == desired_perms)

  # Handle named groups and owning group
  member_count = 0
  perms = 0
  entries = get_acl_entries( path, NAMED_GROUP | OWNING_GROUP )
  for entry in entries:
    if (user_is_member_of_group(user, entry.identity)) :
      member_count += 1
      perms | =  entry.permissions
  if (member_count>0) :
    return ((desired_perms & perms & mask ) == desired_perms)

  # Handle other
  perms = get_perms_for_other(path)
  mask = get_mask( path )
  return ( (desired_perms & perms & mask ) == desired_perms)
```

The mask

As illustrated in the Access Check Algorithm, the mask limits access for **named users**, the **owning group**, and **named groups**.

> **NOTE**
> For a new Data Lake Storage Gen1 account, the mask for the Access ACL of the root folder ("/") defaults to RWX.

The sticky bit

The sticky bit is a more advanced feature of a POSIX filesystem. In the context of Data Lake Storage Gen1, it is unlikely that the sticky bit will be needed. In summary, if the sticky bit is enabled on a folder, a child item can only be deleted or renamed by the child item's owning user.

The sticky bit is not shown in the Azure portal.

Default permissions on new files and folders

When a new file or folder is created under an existing folder, the Default ACL on the parent folder determines:

- A child folder's Default ACL and Access ACL.
- A child file's Access ACL (files do not have a Default ACL).

umask

When creating a file or folder, umask is used to modify how the default ACLs are set on the child item. umask is a 9-bit value on parent folders that contains an RWX value for **owning user**, **owning group**, and **other**.

The umask for Azure Data Lake Storage Gen1 is a constant value set to 007. This value translates to

UMASK COMPONENT	NUMERIC FORM	SHORT FORM	MEANING
umask.owning_user	0	`---`	For owning user, copy the parent's Default ACL to the child's Access ACL
umask.owning_group	0	`---`	For owning group, copy the parent's Default ACL to the child's Access ACL
umask.other	7	`RWX`	For other, remove all permissions on the child's Access ACL

The umask value used by Azure Data Lake Storage Gen1 effectively means that the value for other is never transmitted by default on new children - regardless of what the Default ACL indicates.

The following pseudocode shows how the umask is applied when creating the ACLs for a child item.

```
def set_default_acls_for_new_child(parent, child):
    child.acls = []
    for entry in parent.acls :
        new_entry = None
        if (entry.type == OWNING_USER) :
            new_entry = entry.clone(perms = entry.perms & (~umask.owning_user))
        elif (entry.type == OWNING_GROUP) :
            new_entry = entry.clone(perms = entry.perms & (~umask.owning_group))
        elif (entry.type == OTHER) :
            new_entry = entry.clone(perms = entry.perms & (~umask.other))
        else :
            new_entry = entry.clone(perms = entry.perms )
        child_acls.add( new_entry )
```

Common questions about ACLs in Data Lake Storage Gen1

Do I have to enable support for ACLs?

No. Access control via ACLs is always on for a Data Lake Storage Gen1 account.

Which permissions are required to recursively delete a folder and its contents?

- The parent folder must have **Write + Execute** permissions.
- The folder to be deleted, and every folder within it, requires **Read + Write + Execute** permissions.

> **NOTE**
>
> You do not need Write permissions to delete files in folders. Also, the root folder "/" can **never** be deleted.

Who is the owner of a file or folder?

The creator of a file or folder becomes the owner.

Which group is set as the owning group of a file or folder at creation?

The owning group is copied from the owning group of the parent folder under which the new file or folder is created.

I am the owning user of a file but I don't have the RWX permissions I need. What do I do?

The owning user can change the permissions of the file to give themselves any RWX permissions they need.

When I look at ACLs in the Azure portal I see user names but through APIs, I see GUIDs, why is that?

Entries in the ACLs are stored as GUIDs that correspond to users in Azure AD. The APIs return the GUIDs as is. The Azure portal tries to make ACLs easier to use by translating the GUIDs into friendly names when possible.

Why do I sometimes see GUIDs in the ACLs when I'm using the Azure portal?

A GUID is shown when the user doesn't exist in Azure AD anymore. Usually this happens when the user has left the company or if their account has been deleted in Azure AD. Also, ensure that you're using the right ID for setting ACLs (details in question below).

When using service principal, what ID should I use to set ACLs?

On the Azure Portal, go to **Azure Active Directory -> Enterprise applications** and select your application. The **Overview** tab should display an Object ID and this is what should be used when adding ACLs for data access (and not Application Id).

Does Data Lake Storage Gen1 support inheritance of ACLs?

No, but Default ACLs can be used to set ACLs for child files and folder newly created under the parent folder.

What are the limits for ACL entries on files and folders?

32 ACLs can be set per file and per directory. Access and default ACLs each have their own 32 ACL entry limit. Use security groups for ACL assignments if possible. By using groups, you're less likely to exceed the maximum number of ACL entries per file or directory.

Where can I learn more about POSIX access control model?

- POSIX Access Control Lists on Linux
- HDFS permission guide
- POSIX FAQ
- POSIX 1003.1 2008
- POSIX 1003.1 2013
- POSIX 1003.1 2016
- POSIX ACL on Ubuntu
- ACL using access control lists on Linux

See also

- Overview of Azure Data Lake Storage Gen1

Securing data stored in Azure Data Lake Storage Gen1

10/3/2022 • 8 minutes to read • Edit Online

Securing data in Azure Data Lake Storage Gen1 is a three-step approach. Both Azure role-based access control (Azure RBAC) and access control lists (ACLs) must be set to fully enable access to data for users and security groups.

1. Start by creating security groups in Azure Active Directory (Azure AD). These security groups are used to implement Azure role-based access control (Azure RBAC) in the Azure portal. For more information, see Azure RBAC.
2. Assign the Azure AD security groups to the Data Lake Storage Gen1 account. This controls access to the Data Lake Storage Gen1 account from the portal and management operations from the portal or APIs.
3. Assign the Azure AD security groups as access control lists (ACLs) on the Data Lake Storage Gen1 file system.
4. Additionally, you can also set an IP address range for clients that can access the data in Data Lake Storage Gen1.

This article provides instructions on how to use the Azure portal to perform the above tasks. For in-depth information on how Data Lake Storage Gen1 implements security at the account and data level, see Security in Azure Data Lake Storage Gen1. For deep-dive information on how ACLs are implemented in Data Lake Storage Gen1, see Overview of Access Control in Data Lake Storage Gen1.

Prerequisites

Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **A Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1

Create security groups in Azure Active Directory

For instructions on how to create Azure AD security groups and how to add users to the group, see Managing security groups in Azure Active Directory.

> **NOTE**
>
> You can add both users and other groups to a group in Azure AD using the Azure portal. However, in order to add a service principal to a group, use Azure AD's PowerShell module.

```
# Get the desired group and service principal and identify the correct object IDs
Get-AzureADGroup -SearchString "<group name>"
Get-AzureADServicePrincipal -SearchString "<SPI name>"

# Add the service principal to the group
Add-AzureADGroupMember -ObjectId <Group object ID> -RefObjectId <SPI object ID>
```

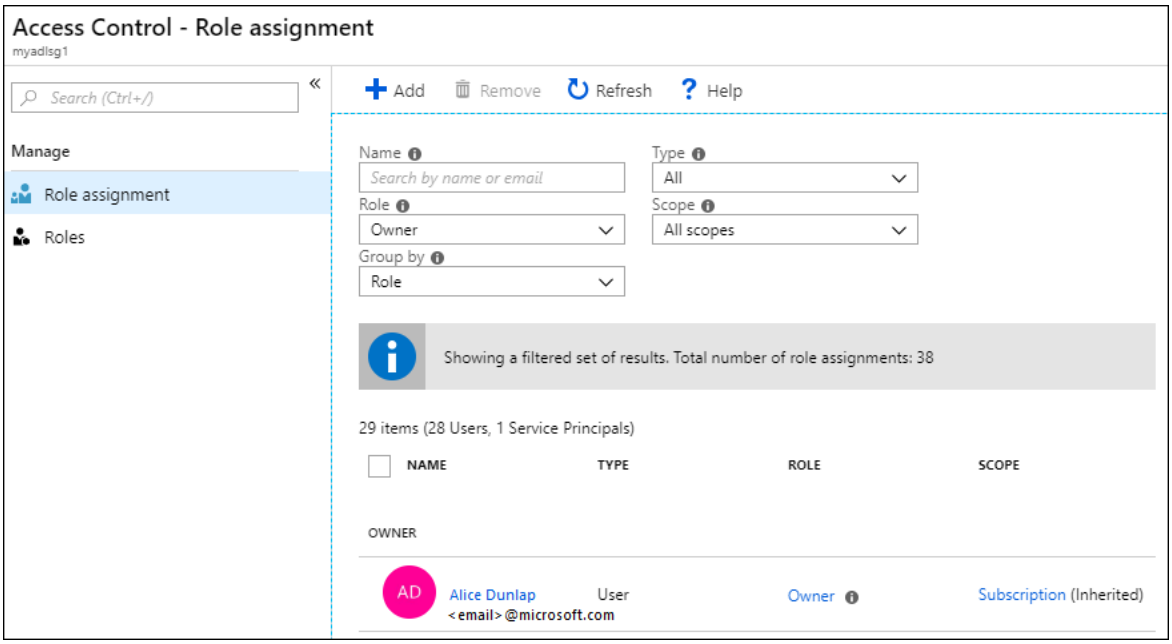
Assign users or security groups to Data Lake Storage Gen1 accounts

When you assign users or security groups to Data Lake Storage Gen1 accounts, you control access to the

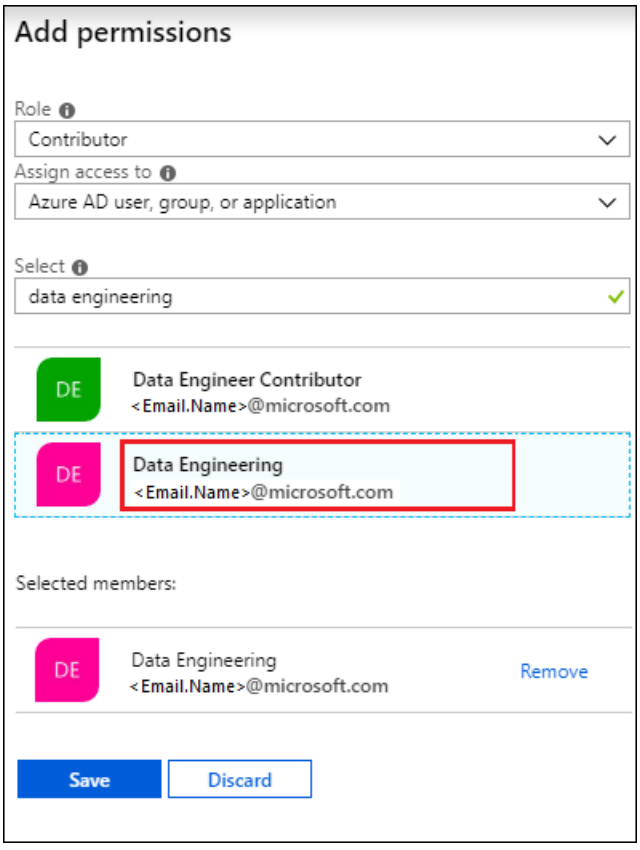
management operations on the account using the Azure portal and Azure Resource Manager APIs.

1. Open a Data Lake Storage Gen1 account. From the left pane, click **All resources**, and then from the All resources blade, click the account name to which you want to assign a user or security group.

2. In your Data Lake Storage Gen1 account blade, click **Access Control (IAM)**. The blade by default lists the subscription owners as the owner.



3. In the **Access Control (IAM)** blade, click **Add** to open the **Add permissions** blade. In the **Add permissions** blade, select a **Role** for the user/group. Look for the security group you created earlier in Azure Active Directory and select it. If you have a lot of users and groups to search from, use the **Select** text box to filter on the group name.



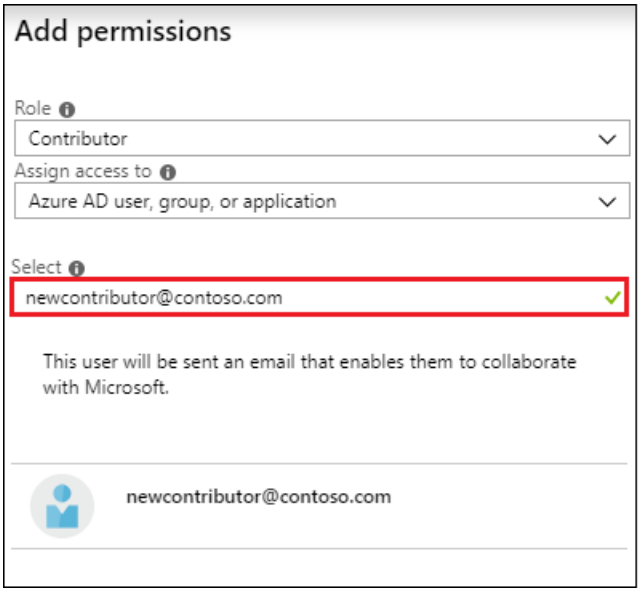
The **Owner** and **Contributor** role provide access to a variety of administration functions on the data lake account. For users who will interact with data in the data lake but still need to view account

management information, you can add them to the **Reader** role. The scope of these roles is limited to the management operations related to the Data Lake Storage Gen1 account.

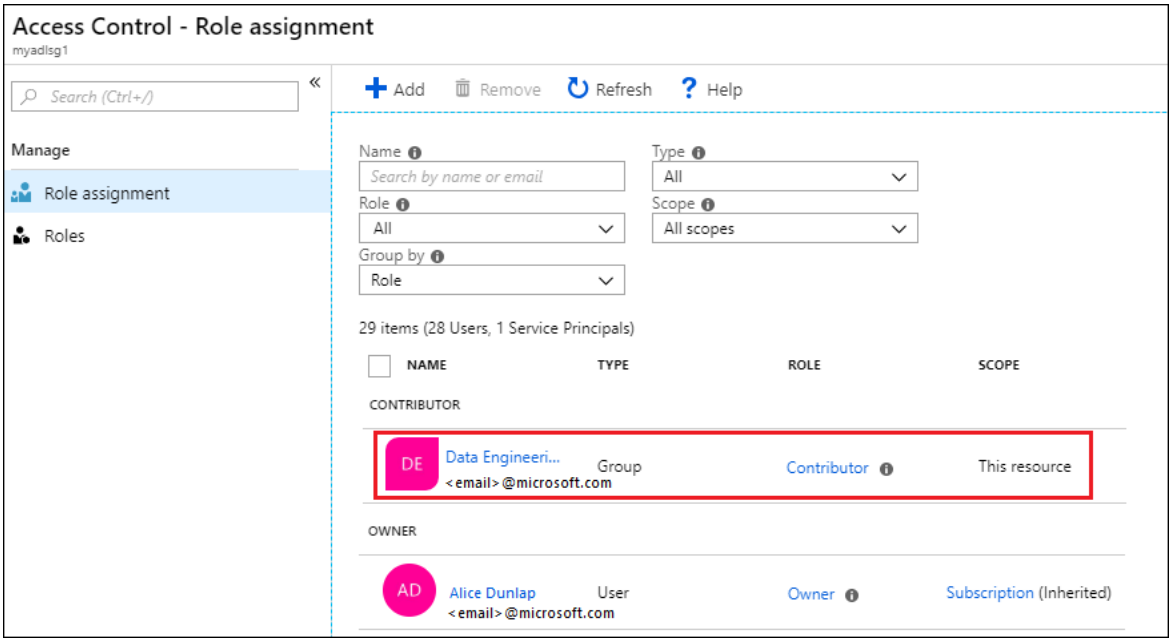
For data operations, individual file system permissions define what the users can do. Therefore, a user having a Reader role can only view administrative settings associated with the account but can potentially read and write data based on file system permissions assigned to them. Data Lake Storage Gen1 file system permissions are described at Assign security group as ACLs to the Azure Data Lake Storage Gen1 file system.

> **IMPORTANT**
>
> Only the **Owner** role automatically enables file system access. The **Contributor**, **Reader**, and all other roles require ACLs to enable any level of access to folders and files. The **Owner** role provides super-user file and folder permissions that cannot be overridden via ACLs. For more information on how Azure RBAC policies map to data access, see Azure RBAC for account management.

4. If you want to add a group/user that is not listed in the **Add permissions** blade, you can invite them by typing their email address in the **Select** text box and then selecting them from the list.



5. Click **Save**. You should see the security group added as shown below.



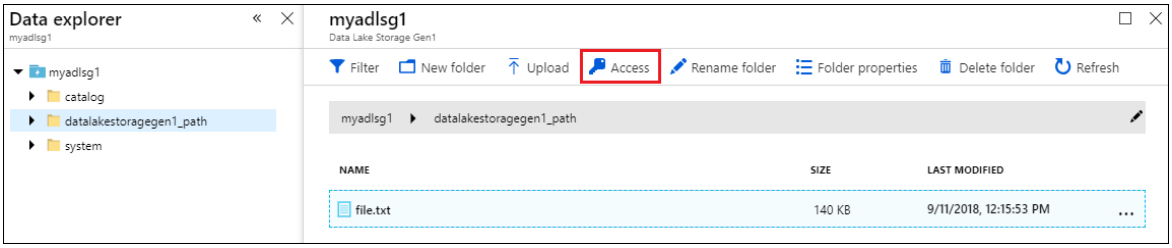
6. Your user/security group now has access to the Data Lake Storage Gen1 account. If you want to provide access to specific users, you can add them to the security group. Similarly, if you want to revoke access for a user, you can remove them from the security group. You can also assign multiple security groups to an account.

Assign users or security groups as ACLs to the Data Lake Storage Gen1 file system

By assigning user/security groups to the Data Lake Storage Gen1 file system, you set access control on the data stored in Data Lake Storage Gen1.

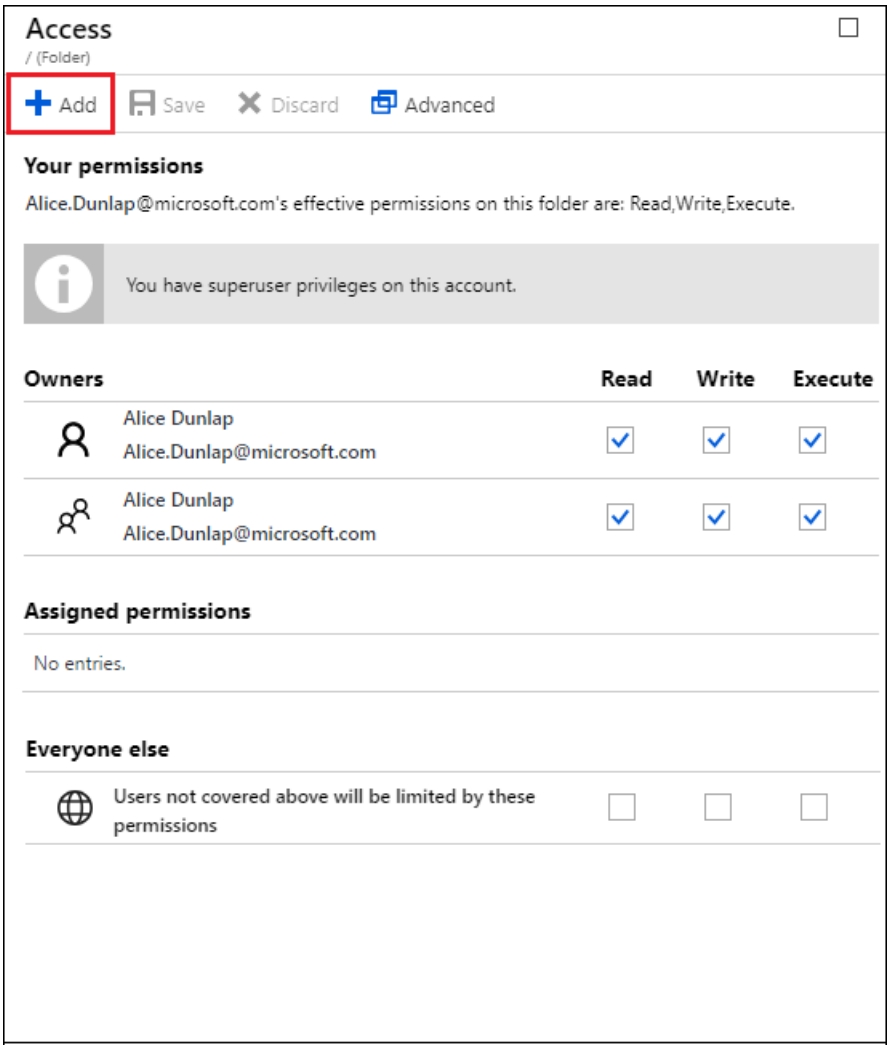
1. In your Data Lake Storage Gen1 account blade, click **Data Explorer**.

2. In the **Data Explorer** blade, click the folder for which you want to configure the ACL, and then click **Access**. To assign ACLs to a file, you must first click the file to preview it and then click **Access** from the **File Preview** blade.



3. The **Access** blade lists the owners and assigned permissions already assigned to the root. Click the **Add** icon to add additional Access ACLs.

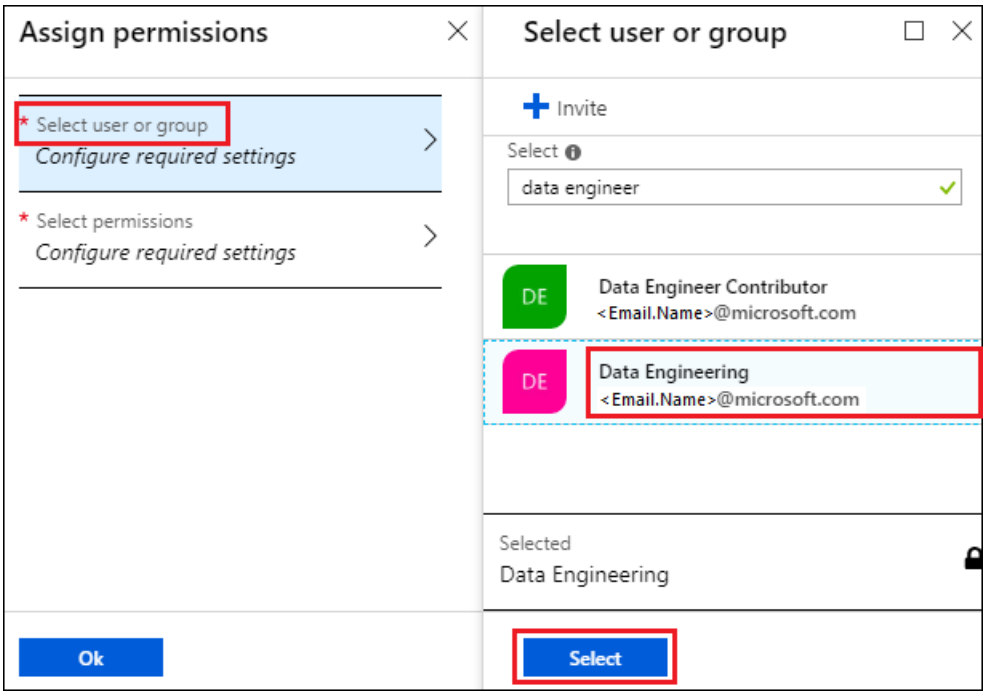
> **IMPORTANT**
>
> Setting access permissions for a single file does not necessarily grant a user/group access to that file. The path to the file must be accessible to the assigned user/group. For more information and examples, see Common scenarios related to permissions.



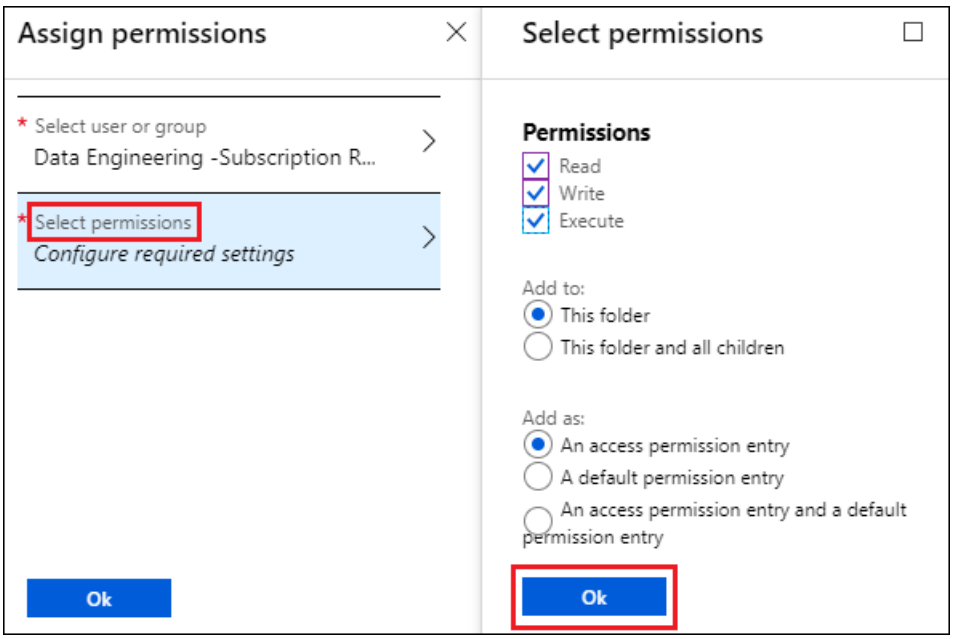
- The **Owners** and **Everyone else** provide UNIX-style access, where you specify read, write, execute (rwx) to three distinct user classes: owner, group, and others.
- **Assigned permissions** corresponds to the POSIX ACLs that enable you to set permissions for specific named users or groups beyond the file's owner or group.

  For more information, see HDFS ACLs. For more information on how ACLs are implemented in Data Lake Storage Gen1, see Access Control in Data Lake Storage Gen1.

4. Click the **Add** icon to open the **Assign permissions** blade. In this blade, click **Select user or group**, and then in **Select user or group** blade, look for the security group you created earlier in Azure Active Directory. If you have a lot of groups to search from, use the text box at the top to filter on the group name. Click the group you want to add and then click **Select**.

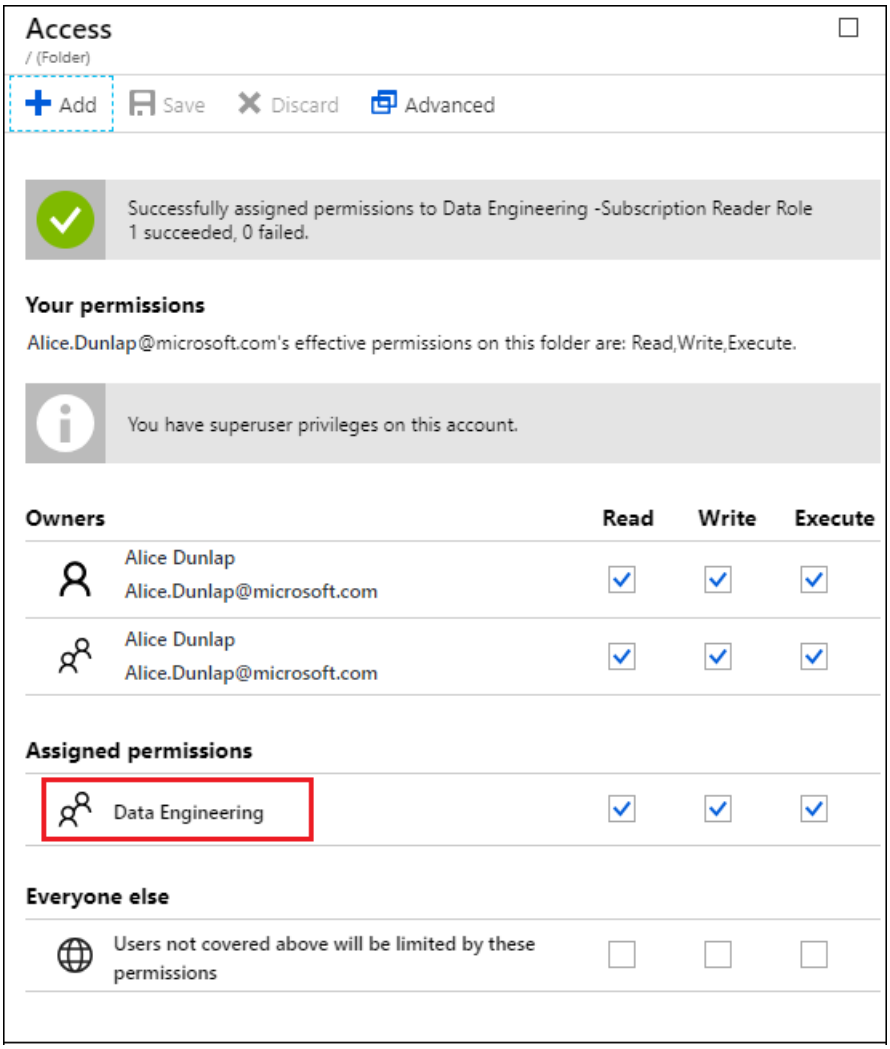


5. Click **Select permissions**, select the permissions, whether the permissions should be applied to recursively, and whether you want to assign the permissions as an access ACL, default ACL, or both. Click **OK**.



For more information about permissions in Data Lake Storage Gen1, and Default/Access ACLs, see Access Control in Data Lake Storage Gen1.

6. After clicking **Ok** in the **Select permissions** blade, the newly added group and associated permissions will now be listed in the **Access** blade.



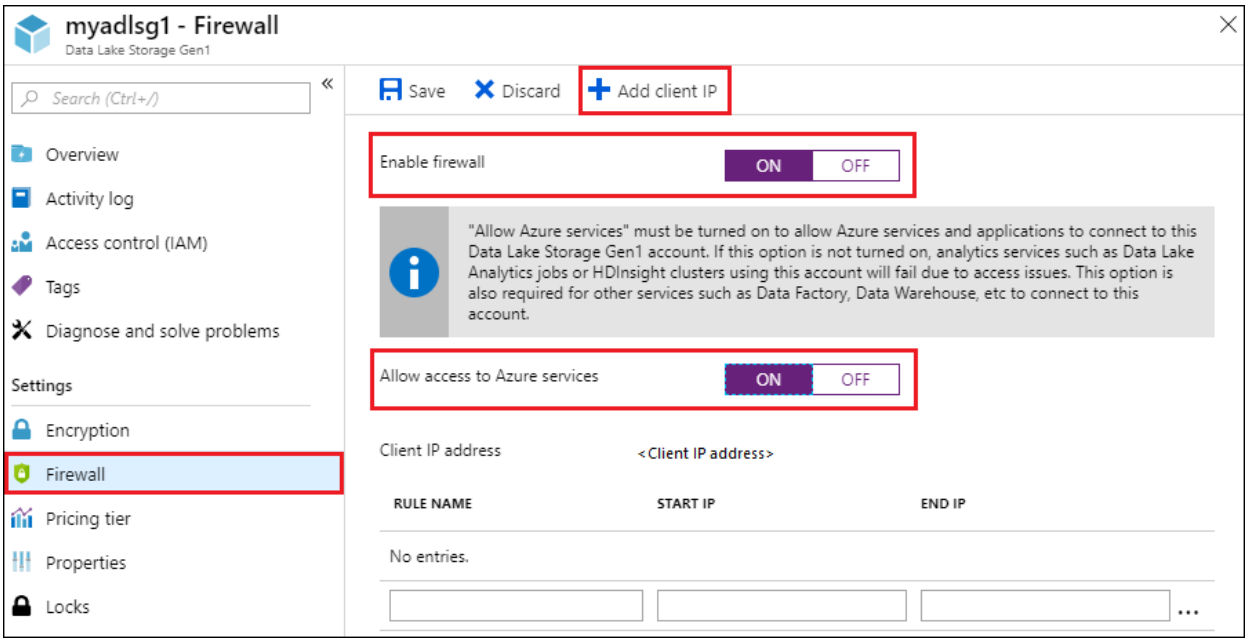
IMPORTANT

In the current release, you can have up to 28 entries under **Assigned permissions**. If you want to add more than 28 users, you should create security groups, add users to security groups, add provide access to those security groups for the Data Lake Storage Gen1 account.

7. If required, you can also modify the access permissions after you have added the group. Clear or select the check box for each permission type (Read, Write, Execute) based on whether you want to remove or assign that permission to the security group. Click **Save** to save the changes, or **Discard** to undo the changes.

Set IP address range for data access

Data Lake Storage Gen1 enables you to further lock down access to your data store at network level. You can enable firewall, specify an IP address, or define an IP address range for your trusted clients. Once enabled, only clients that have the IP addresses within defined range can connect to the store.

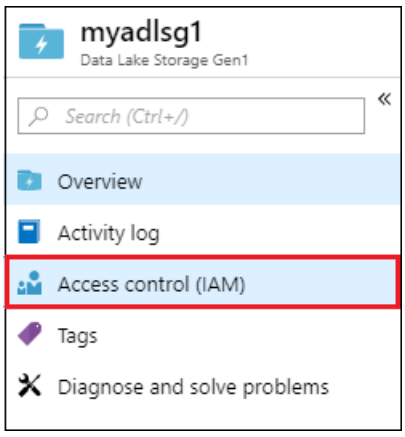


Remove security groups for a Data Lake Storage Gen1 account

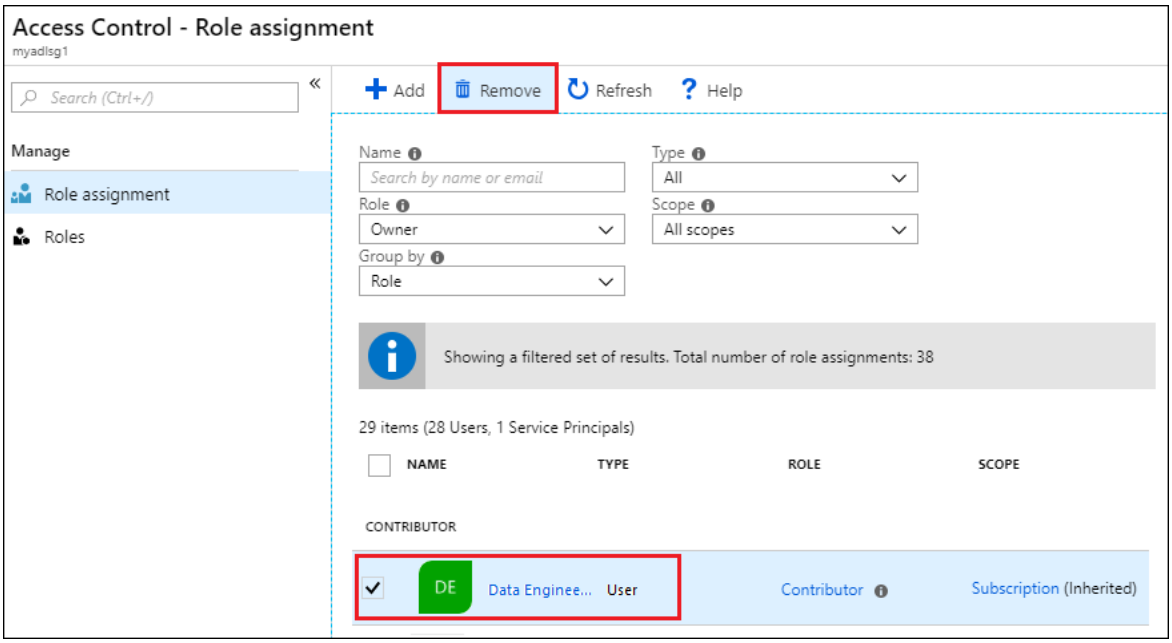
When you remove security groups from Data Lake Storage Gen1 accounts, you are only changing access to the management operations on the account using the Azure portal and Azure Resource Manager APIs.

Access to data is unchanged and is still managed by the access ACLs. The exception to this are users/groups in the Owners role. Users/groups removed from the Owners role are no longer super users and their access falls back to access ACL settings.

1. In your Data Lake Storage Gen1 account blade, click **Access Control (IAM)**.



2. In the **Access Control (IAM)** blade, click the security group(s) you want to remove. Click **Remove**.

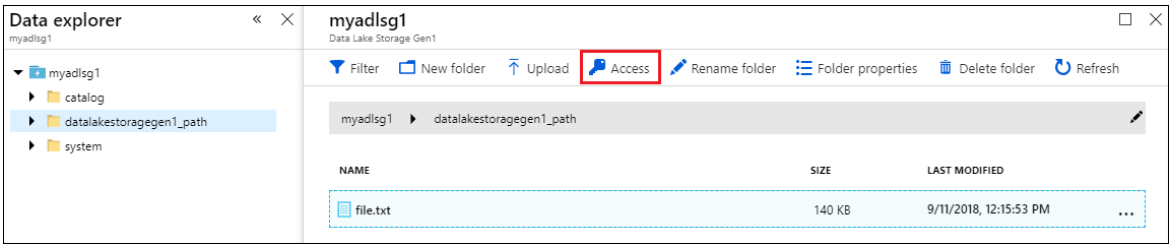


Remove security group ACLs from a Data Lake Storage Gen1 file system

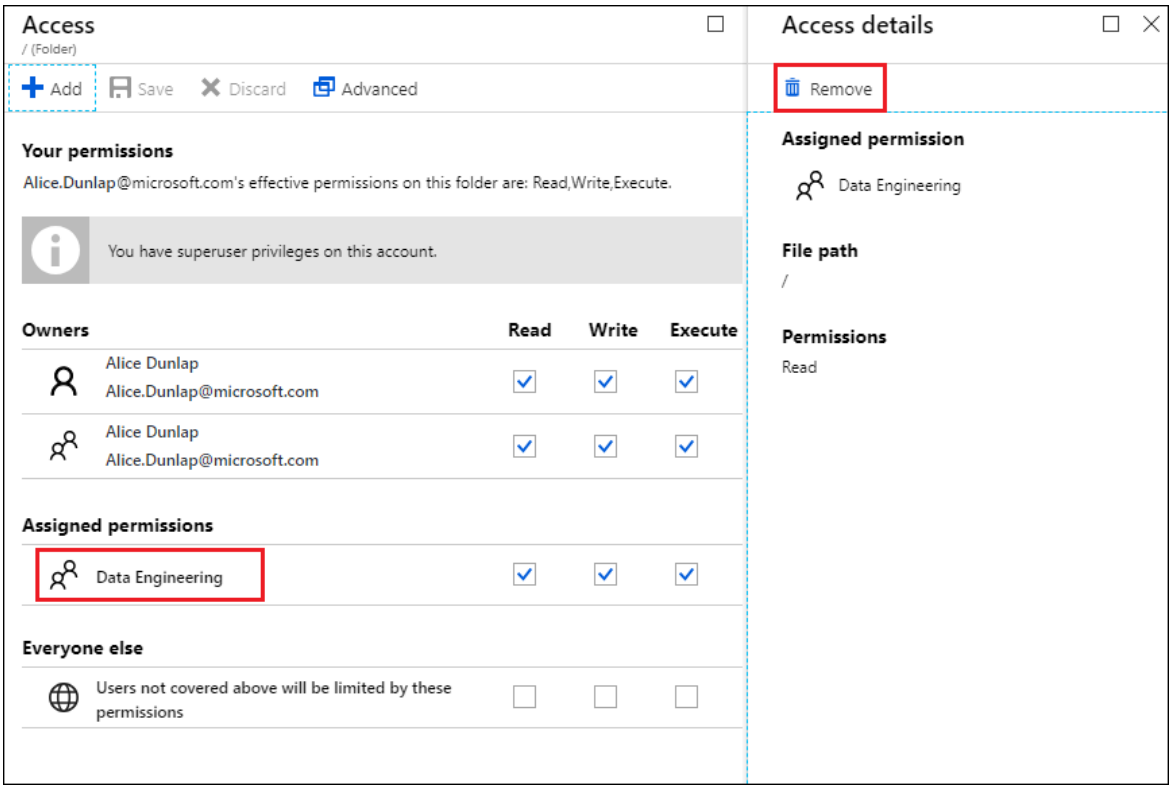
When you remove security group ACLs from a Data Lake Storage Gen1 file system, you change access to the data in the Data Lake Storage Gen1 account.

1. In your Data Lake Storage Gen1 account blade, click **Data Explorer**.

2. In the **Data Explorer** blade, click the folder for which you want to remove the ACL, and then click **Access**. To remove ACLs for a file, you must first click the file to preview it and then click **Access** from the **File Preview** blade.



3. In the **Access** blade, click the security group you want to remove. In the **Access details** blade, click **Remove**.



See also

- Overview of Azure Data Lake Storage Gen1
- Copy data from Azure Storage Blobs to Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1
- Get Started with Data Lake Storage Gen1 using PowerShell
- Get Started with Data Lake Storage Gen1 using .NET SDK
- Access diagnostic logs for Data Lake Storage Gen1

Encryption of data in Azure Data Lake Storage Gen1

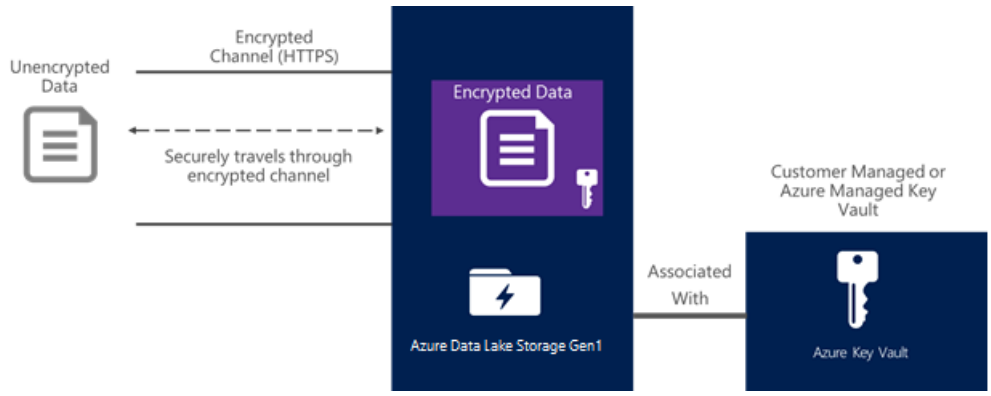
10/3/2022 • 7 minutes to read • Edit Online

Encryption in Azure Data Lake Storage Gen1 helps you protect your data, implement enterprise security policies, and meet regulatory compliance requirements. This article provides an overview of the design, and discusses some of the technical aspects of implementation.

Data Lake Storage Gen1 supports encryption of data both at rest and in transit. For data at rest, Data Lake Storage Gen1 supports "on by default," transparent encryption. Here is what these terms mean in a bit more detail:

- **On by default**: When you create a new Data Lake Storage Gen1 account, the default setting enables encryption. Thereafter, data that is stored in Data Lake Storage Gen1 is always encrypted prior to storing on persistent media. This is the behavior for all data, and it cannot be changed after an account is created.
- **Transparent**: Data Lake Storage Gen1 automatically encrypts data prior to persisting, and decrypts data prior to retrieval. The encryption is configured and managed at the Data Lake Storage Gen1 account level by an administrator. No changes are made to the data access APIs. Thus, no changes are required in applications and services that interact with Data Lake Storage Gen1 because of encryption.

Data in transit (also known as data in motion) is also always encrypted in Data Lake Storage Gen1. In addition to encrypting data prior to storing to persistent media, the data is also always secured in transit by using HTTPS. HTTPS is the only protocol that is supported for the Data Lake Storage Gen1 REST interfaces. The following diagram shows how data becomes encrypted in Data Lake Storage Gen1:



Set up encryption with Data Lake Storage Gen1

Encryption for Data Lake Storage Gen1 is set up during account creation, and it is always enabled by default. You can either manage the keys yourself, or allow Data Lake Storage Gen1 to manage them for you (this is the default).

For more information, see Getting started.

How encryption works in Data Lake Storage Gen1

The following information covers how to manage master encryption keys, and it explains the three different types of keys you can use in data encryption for Data Lake Storage Gen1.

Master encryption keys

Data Lake Storage Gen1 provides two modes for management of master encryption keys (MEKs). For now, assume that the master encryption key is the top-level key. Access to the master encryption key is required to decrypt any data stored in Data Lake Storage Gen1.

The two modes for managing the master encryption key are as follows:

- Service managed keys
- Customer managed keys

In both modes, the master encryption key is secured by storing it in Azure Key Vault. Key Vault is a fully managed, highly secure service on Azure that can be used to safeguard cryptographic keys. For more information, see Key Vault.

Here is a brief comparison of capabilities provided by the two modes of managing the MEKs.

QUESTION	SERVICE MANAGED KEYS	CUSTOMER MANAGED KEYS
How is data stored?	Always encrypted prior to being stored.	Always encrypted prior to being stored.
Where is the Master Encryption Key stored?	Key Vault	Key Vault
Are any encryption keys stored in the clear outside of Key Vault?	No	No
Can the MEK be retrieved by Key Vault?	No. After the MEK is stored in Key Vault, it can only be used for encryption and decryption.	No. After the MEK is stored in Key Vault, it can only be used for encryption and decryption.
Who owns the Key Vault instance and the MEK?	The Data Lake Storage Gen1 service	You own the Key Vault instance, which belongs in your own Azure subscription. The MEK in Key Vault can be managed by software or hardware.
Can you revoke access to the MEK for the Data Lake Storage Gen1 service?	No	Yes. You can manage access control lists in Key Vault, and remove access control entries to the service identity for the Data Lake Storage Gen1 service.
Can you permanently delete the MEK?	No	Yes. If you delete the MEK from Key Vault, the data in the Data Lake Storage Gen1 account cannot be decrypted by anyone, including the Data Lake Storage Gen1 service. If you have explicitly backed up the MEK prior to deleting it from Key Vault, the MEK can be restored, and the data can then be recovered. However, if you have not backed up the MEK prior to deleting it from Key Vault, the data in the Data Lake Storage Gen1 account can never be decrypted thereafter.

Aside from this difference of who manages the MEK and the Key Vault instance in which it resides, the rest of the design is the same for both modes.

It's important to remember the following when you choose the mode for the master encryption keys:

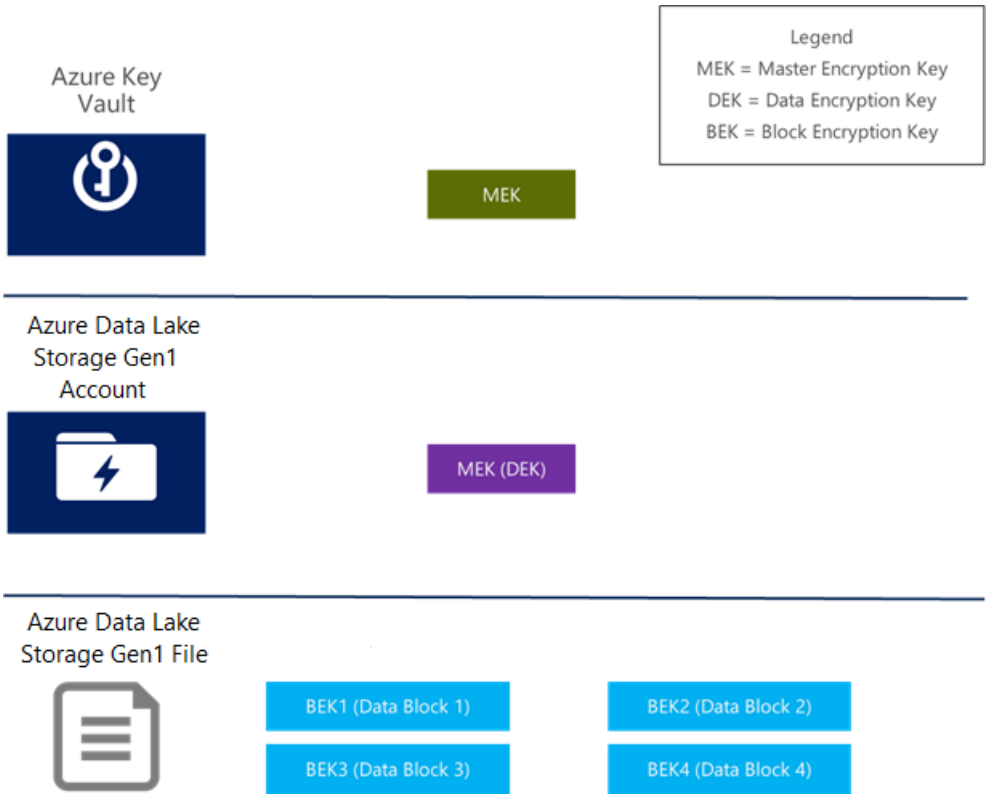
- You can choose whether to use customer managed keys or service managed keys when you provision a Data Lake Storage Gen1 account.
- After a Data Lake Storage Gen1 account is provisioned, the mode cannot be changed.

Encryption and decryption of data

There are three types of keys that are used in the design of data encryption. The following table provides a summary:

KEY	ABBREVIATION	ASSOCIATED WITH	STORAGE LOCATION	TYPE	NOTES
Master Encryption Key	MEK	A Data Lake Storage Gen1 account	Key Vault	Asymmetric	It can be managed by Data Lake Storage Gen1 or you.
Data Encryption Key	DEK	A Data Lake Storage Gen1 account	Persistent storage, managed by the Data Lake Storage Gen1 service	Symmetric	The DEK is encrypted by the MEK. The encrypted DEK is what is stored on persistent media.
Block Encryption Key	BEK	A block of data	None	Symmetric	The BEK is derived from the DEK and the data block.

The following diagram illustrates these concepts:



Pseudo algorithm when a file is to be decrypted:

1. Check if the DEK for the Data Lake Storage Gen1 account is cached and ready for use.
   - If not, then read the encrypted DEK from persistent storage, and send it to Key Vault to be decrypted.

Cache the decrypted DEK in memory. It is now ready to use.

2. For every block of data in the file:
   - Read the encrypted block of data from persistent storage.
   - Generate the BEK from the DEK and the encrypted block of data.
   - Use the BEK to decrypt data.

Pseudo algorithm when a block of data is to be encrypted:

1. Check if the DEK for the Data Lake Storage Gen1 account is cached and ready for use.
   - If not, then read the encrypted DEK from persistent storage, and send it to Key Vault to be decrypted. Cache the decrypted DEK in memory. It is now ready to use.
2. Generate a unique BEK for the block of data from the DEK.
3. Encrypt the data block with the BEK, by using AES-256 encryption.
4. Store the encrypted data block of data on persistent storage.

> **NOTE**
> The DEK is always stored encrypted by the MEK, whether on persistent media or cached in memory.

Key rotation

When you are using customer-managed keys, you can rotate the MEK. To learn how to set up a Data Lake Storage Gen1 account with customer-managed keys, see Getting started.

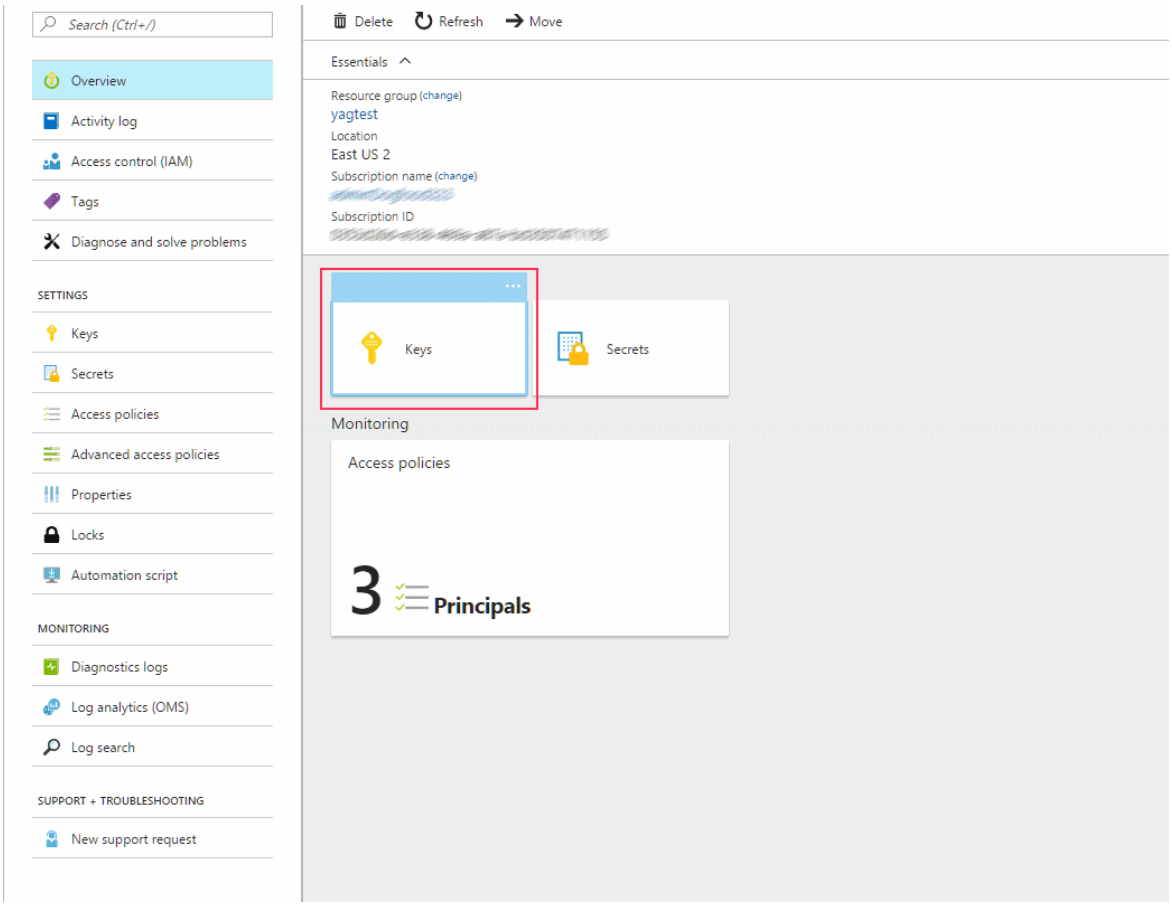
Prerequisites

When you set up the Data Lake Storage Gen1 account, you have chosen to use your own keys. This option cannot be changed after the account has been created. The following steps assume that you are using customer-managed keys (that is, you have chosen your own keys from Key Vault).

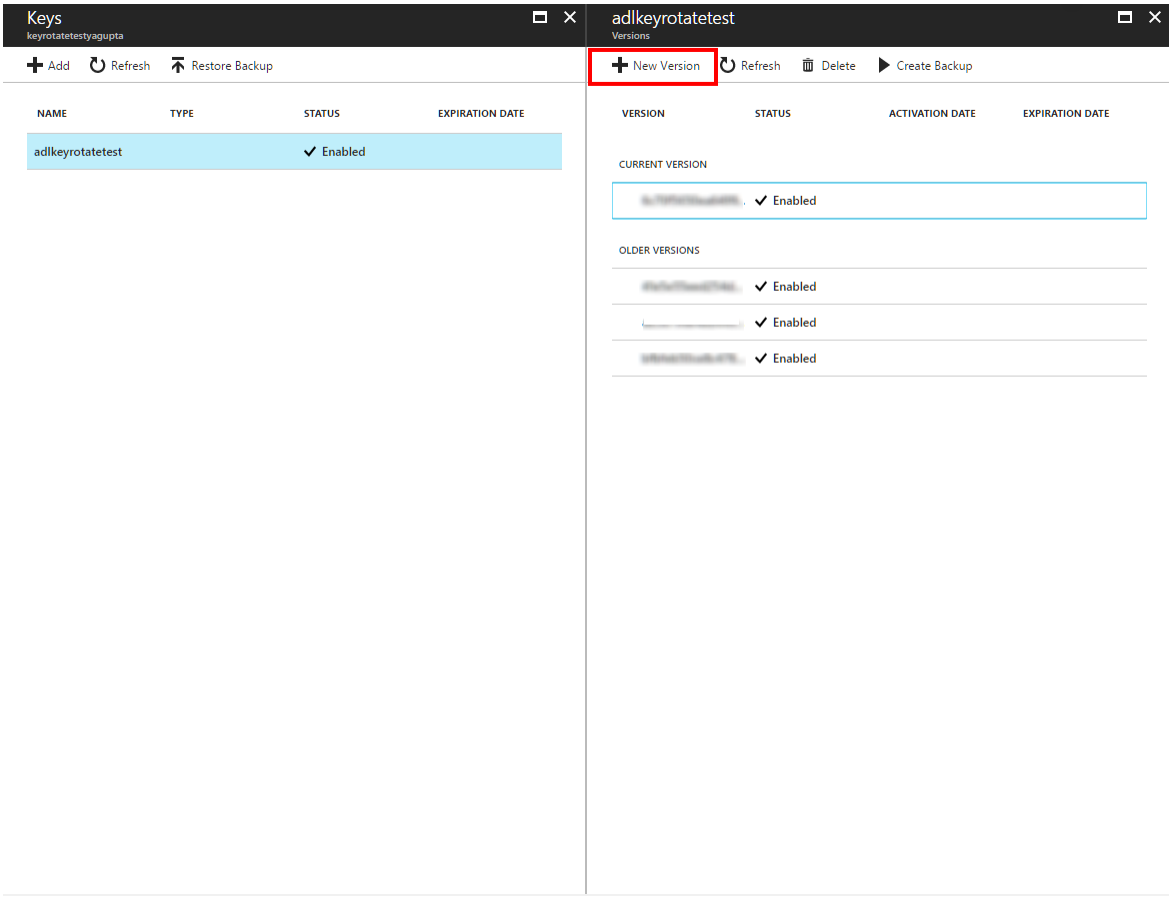
Note that if you use the default options for encryption, your data is always encrypted by using keys managed by Data Lake Storage Gen1. In this option, you don't have the ability to rotate keys, as they are managed by Data Lake Storage Gen1.

How to rotate the MEK in Data Lake Storage Gen1

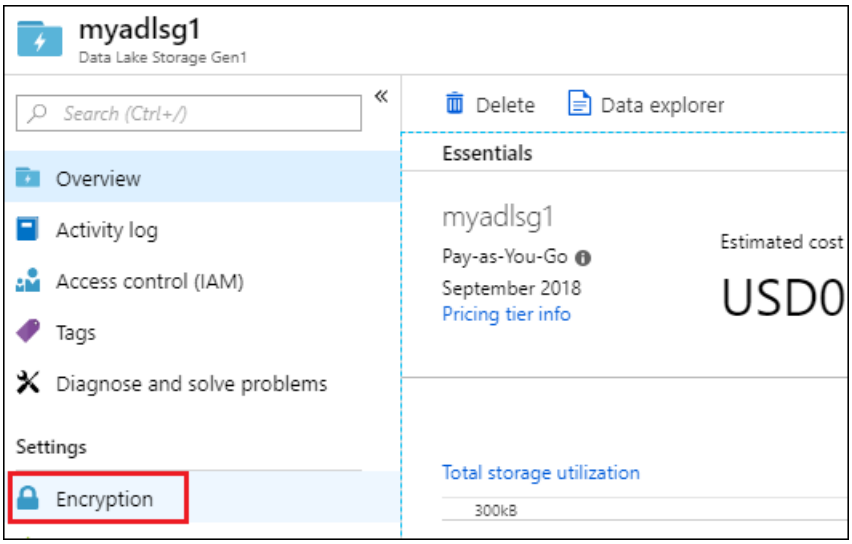
1. Sign in to the Azure portal.
2. Browse to the Key Vault instance that stores your keys associated with your Data Lake Storage Gen1 account. Select **Keys**.



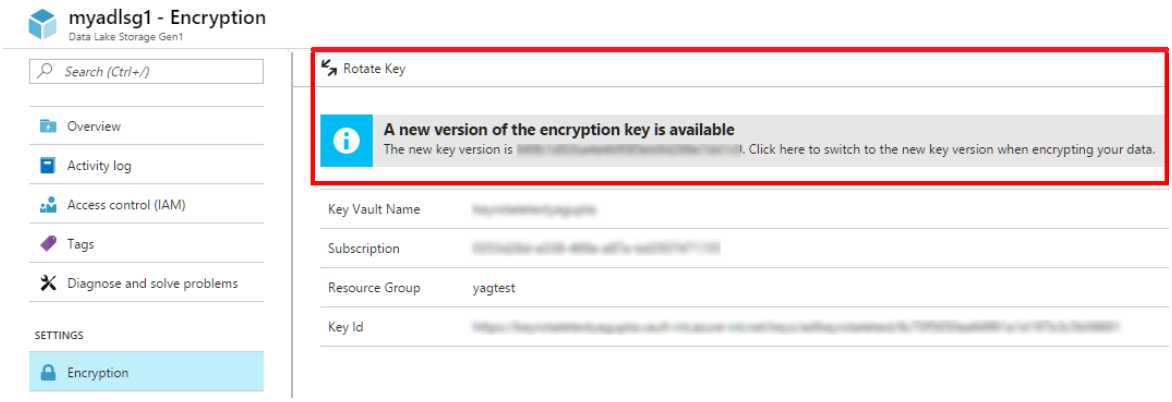
3. Select the key associated with your Data Lake Storage Gen1 account, and create a new version of this key. Note that Data Lake Storage Gen1 currently only supports key rotation to a new version of a key. It doesn't support rotating to a different key.



4. Browse to the Data Lake Storage Gen1 account, and select **Encryption**.



5. A message notifies you that a new key version of the key is available. Click **Rotate Key** to update the key to the new version.



This operation should take less than two minutes, and there is no expected downtime due to key rotation. After the operation is complete, the new version of the key is in use.

> **IMPORTANT**
>
> After the key rotation operation is complete, the old version of the key is no longer actively used for encrypting new data. There may be cases however where accessing older data may need the old key. To allow for reading of such older data, do not delete the old key

Virtual network integration for Azure Data Lake Storage Gen1

10/3/2022 • 6 minutes to read • Edit Online

This article introduces virtual network integration for Azure Data Lake Storage Gen1. With virtual network integration, you can configure your accounts to accept traffic only from specific virtual networks and subnets.

This feature helps to secure your Data Lake Storage account from external threats.

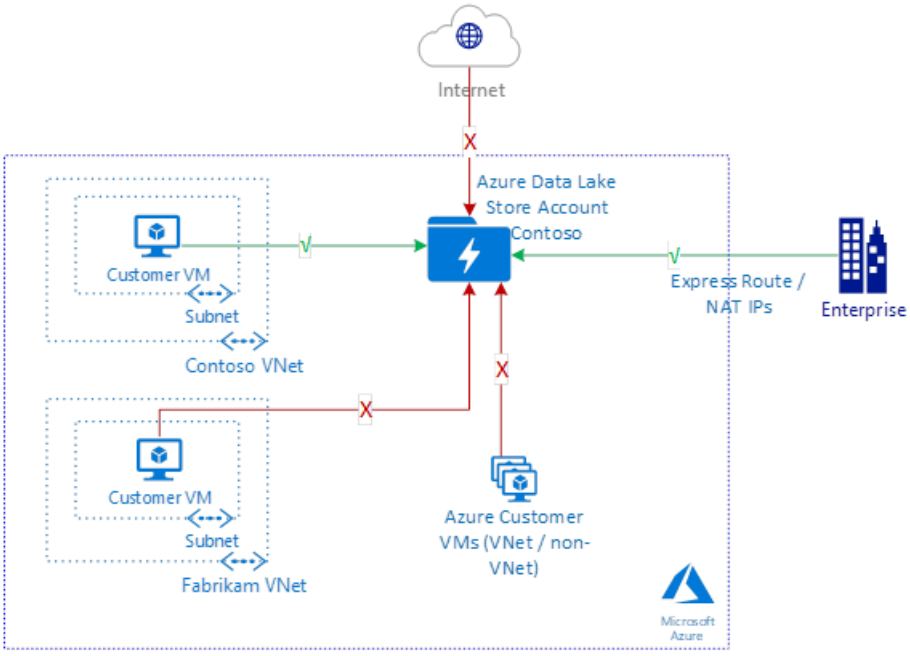
Virtual network integration for Data Lake Storage Gen1 makes use of the virtual network service endpoint security between your virtual network and Azure Active Directory (Azure AD) to generate additional security claims in the access token. These claims are then used to authenticate your virtual network to your Data Lake Storage Gen1 account and allow access.

NOTE

There's no additional charge associated with using these capabilities. Your account is billed at the standard rate for Data Lake Storage Gen1. For more information, see pricing. For all other Azure services that you use, see pricing.

Scenarios for virtual network integration for Data Lake Storage Gen1

With Data Lake Storage Gen1 virtual network integration, you can restrict access to your Data Lake Storage Gen1 account from specific virtual networks and subnets. After your account is locked to the specified virtual network subnet, other virtual networks/VMs in Azure aren't allowed access. Functionally, Data Lake Storage Gen1 virtual network integration enables the same scenario as virtual network service endpoints. A few key differences are detailed in the following sections.



NOTE

The existing IP firewall rules can be used in addition to virtual network rules to allow access from on-premises networks too.

Optimal routing with Data Lake Storage Gen1 virtual network integration

A key benefit of virtual network service endpoints is optimal routing from your virtual network. You can perform the same route optimization to Data Lake Storage Gen1 accounts. Use the following user-defined routes from your virtual network to your Data Lake Storage Gen1 account.

Data Lake Storage public IP address – Use the public IP address for your target Data Lake Storage Gen1 accounts. To identify the IP addresses for your Data Lake Storage Gen1 account, resolve the DNS names of your accounts. Create a separate entry for each address.

```
# Create a route table for your resource group.
az network route-table create --resource-group $RgName --name $RouteTableName

# Create route table rules for Data Lake Storage public IP addresses.
# There's one rule per Data Lake Storage public IP address.
az network route-table route create --name toADLSregion1 --resource-group $RgName --route-table-name
$RouteTableName --address-prefix <ADLS Public IP Address> --next-hop-type Internet

# Update the virtual network, and apply the newly created route table to it.
az network vnet subnet update --vnet-name $VnetName --name $SubnetName --resource-group $RgName --route-
table $RouteTableName
```

Data exfiltration from the customer virtual network

In addition to securing the Data Lake Storage accounts for access from the virtual network, you also might be interested in making sure there's no exfiltration to an unauthorized account.

Use a firewall solution in your virtual network to filter the outbound traffic based on the destination account URL. Allow access to only approved Data Lake Storage Gen1 accounts.

Some available options are:

- Azure Firewall: Deploy and configure an Azure firewall for your virtual network. Secure the outbound Data Lake Storage traffic, and lock it down to the known and approved account URL.
- Network virtual appliance firewall: Your administrator might allow the use of only certain commercial firewall vendors. Use a network virtual appliance firewall solution that's available in the Azure Marketplace to perform the same function.

> **NOTE**
> Using firewalls in the data path introduces an additional hop in the data path. It might affect the network performance for end-to-end data exchange. Throughput availability and connection latency might be affected.

Limitations

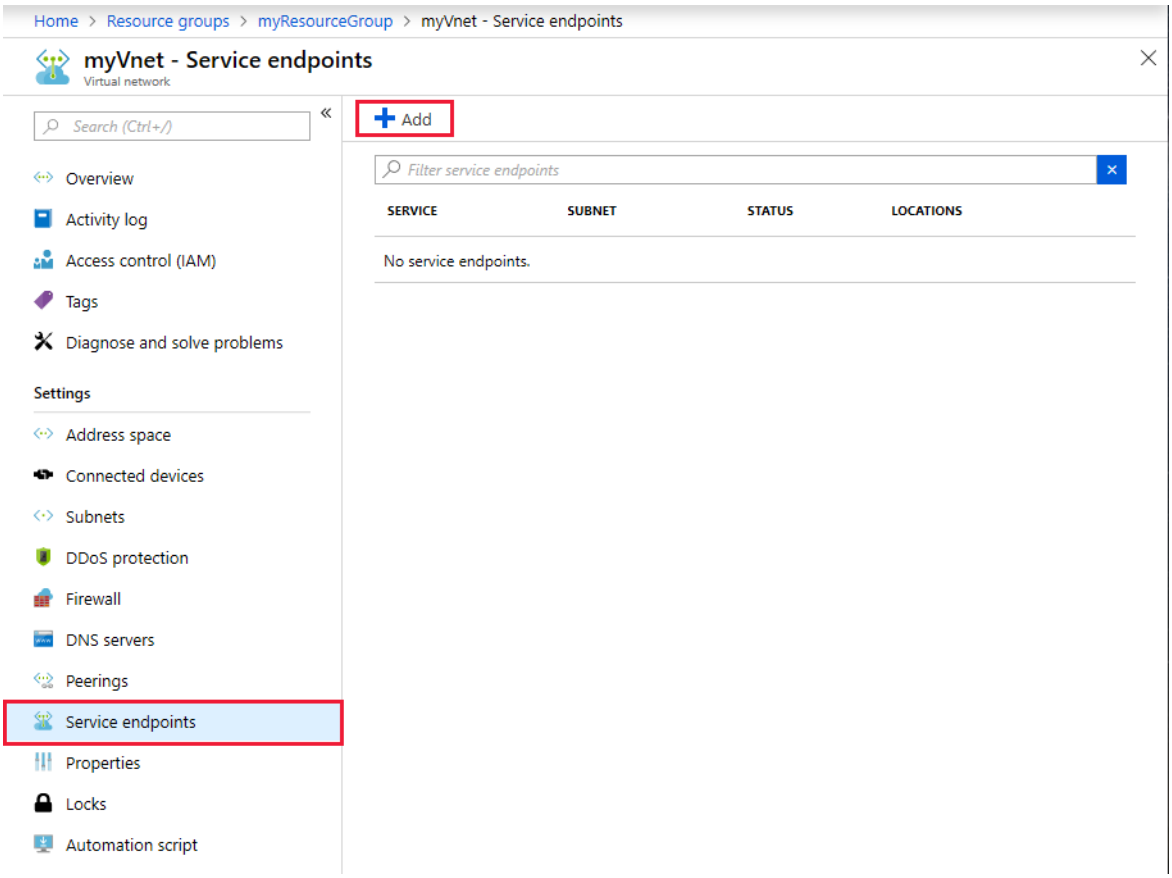
- HDInsight clusters that were created before Data Lake Storage Gen1 virtual network integration support was available must be re-created to support this new feature.
- When you create a new HDInsight cluster and select a Data Lake Storage Gen1 account with virtual network integration enabled, the process fails. First, disable the virtual network rule. Or on the **Firewall and virtual networks** blade of the Data Lake Storage account, select **Allow access from all networks and services**. Then create the HDInsight cluster before finally re-enabling the virtual network rule or de-selecting **Allow access from all networks and services**. For more information, see the Exceptions section.

- Data Lake Storage Gen1 virtual network integration doesn't work with managed identities for Azure resources.
- File and folder data in your virtual network-enabled Data Lake Storage Gen1 account isn't accessible from the portal. This restriction includes access from a VM that's within the virtual network and activities such as using Data Explorer. Account management activities continue to work. File and folder data in your virtual network-enabled Data Lake Storage account is accessible via all non-portal resources. These resources include SDK access, PowerShell scripts, and other Azure services when they don't originate from the portal.

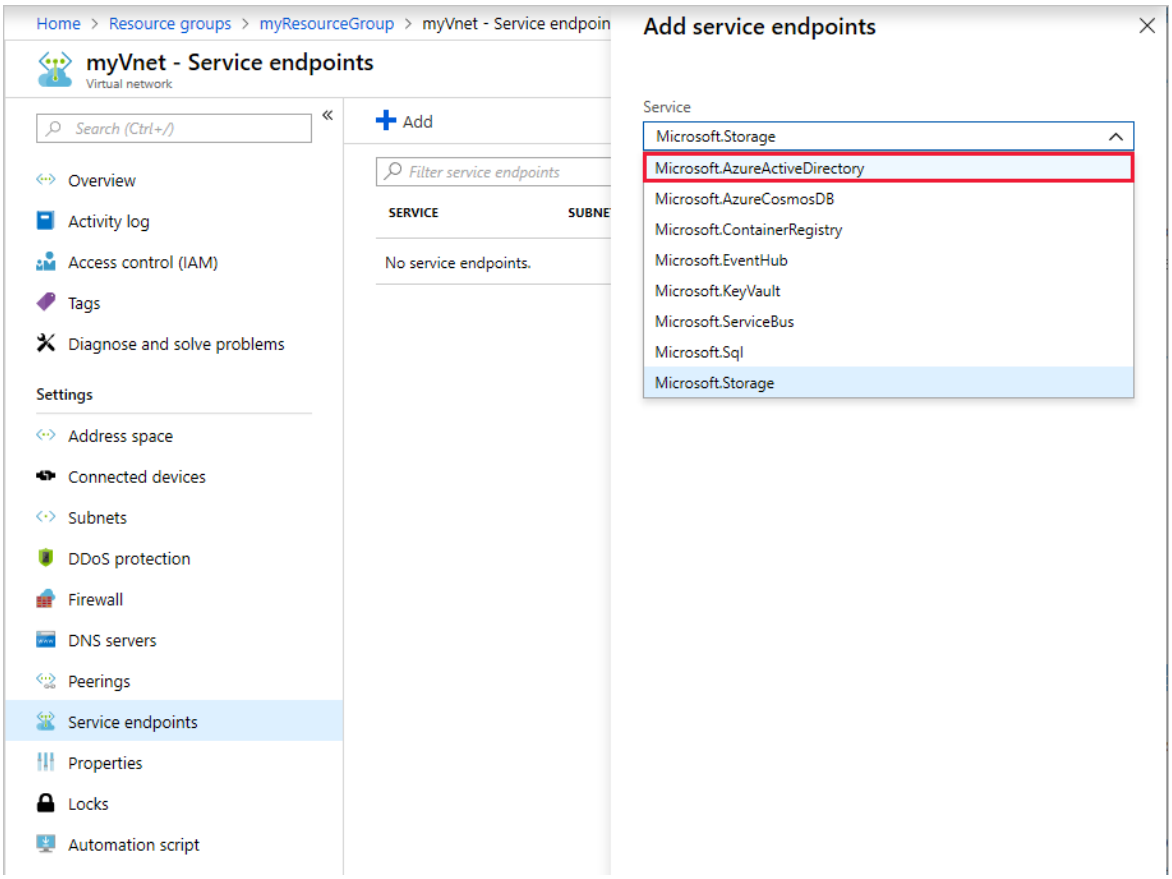
Configuration

Step 1: Configure your virtual network to use an Azure AD service endpoint

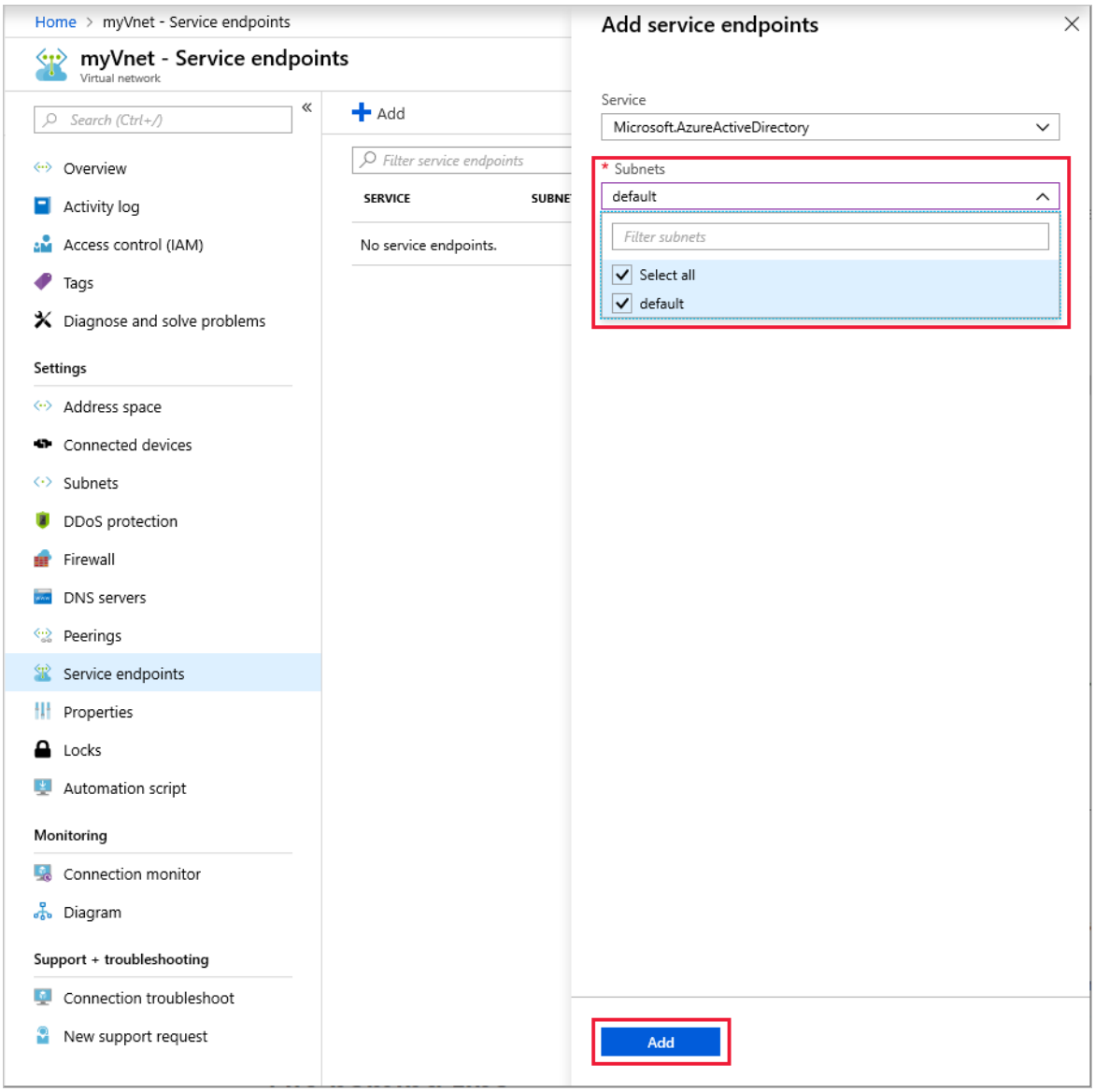
1. Go to the Azure portal, and sign in to your account.
2. Create a new virtual networkin your subscription. Or you can go to an existing virtual network. The virtual network must be in the same region as the Data Lake Storage Gen 1 account.
3. On the **Virtual network** blade, select **Service endpoints**.
4. Select **Add** to add a new service endpoint.



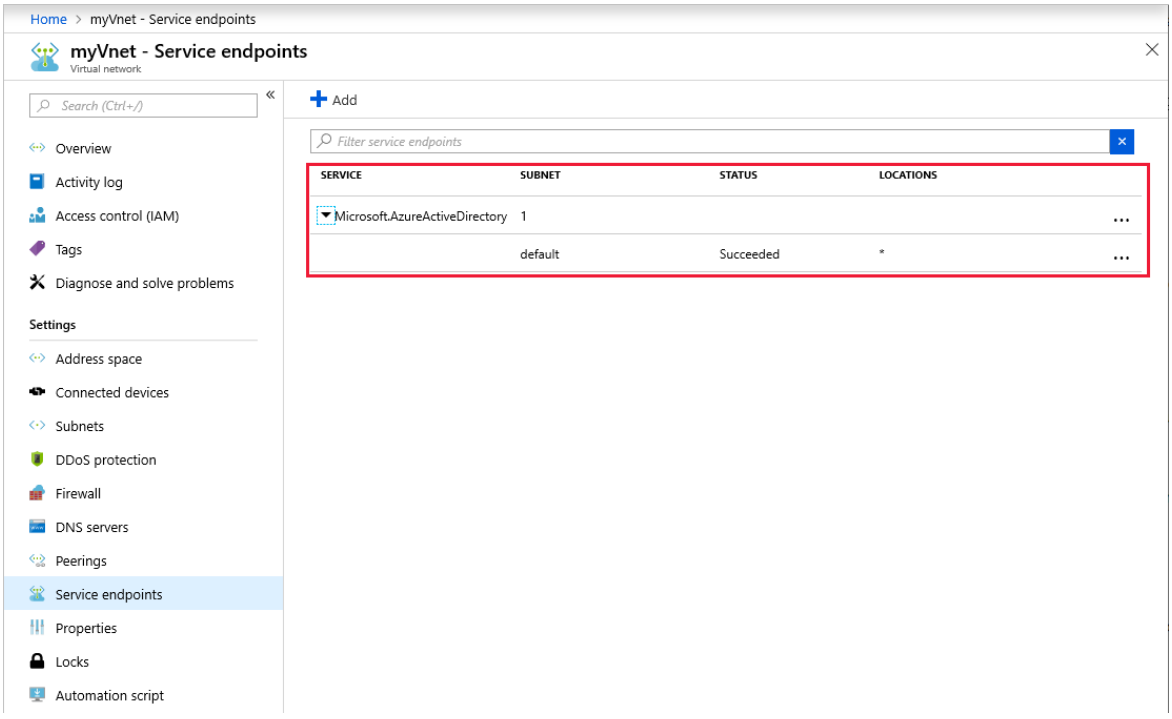
5. Select **Microsoft.AzureActiveDirectory** as the service for the endpoint.



6. Select the subnets for which you intend to allow connectivity. Select **Add**.



7. It can take up to 15 minutes for the service endpoint to be added. After it's added, it shows up in the list. Verify that it shows up and that all details are as configured.

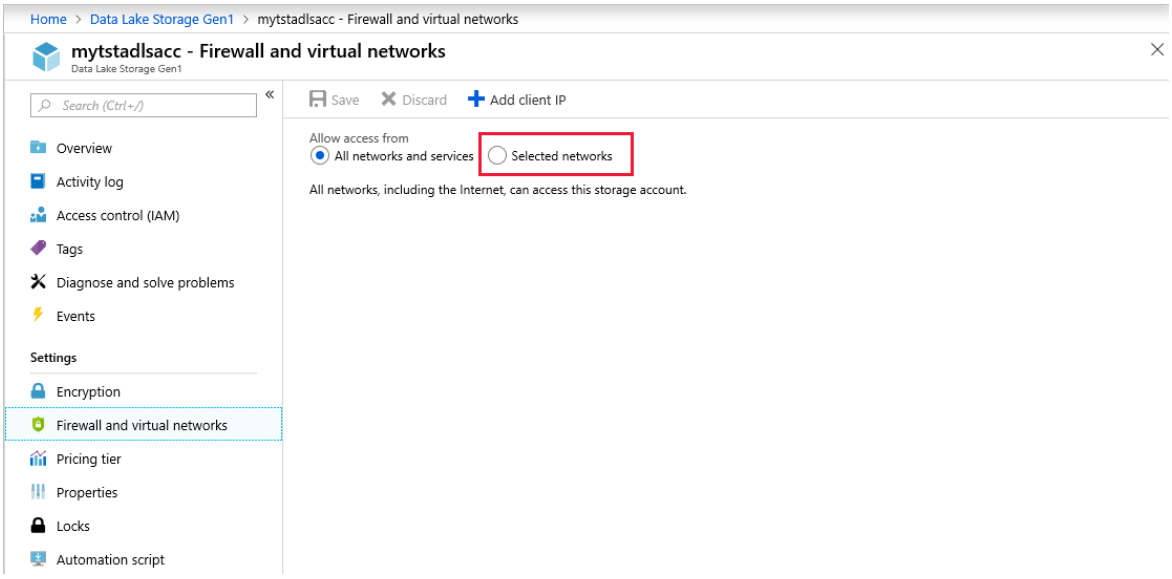


Step 2: Set up the allowed virtual network or subnet for your Data Lake Storage Gen1 account

1. After you configure your virtual network, create a new Azure Data Lake Storage Gen1 account in your subscription. Or you can go to an existing Data Lake Storage Gen1 account. The Data Lake Storage Gen1 account must be in the same region as the virtual network.

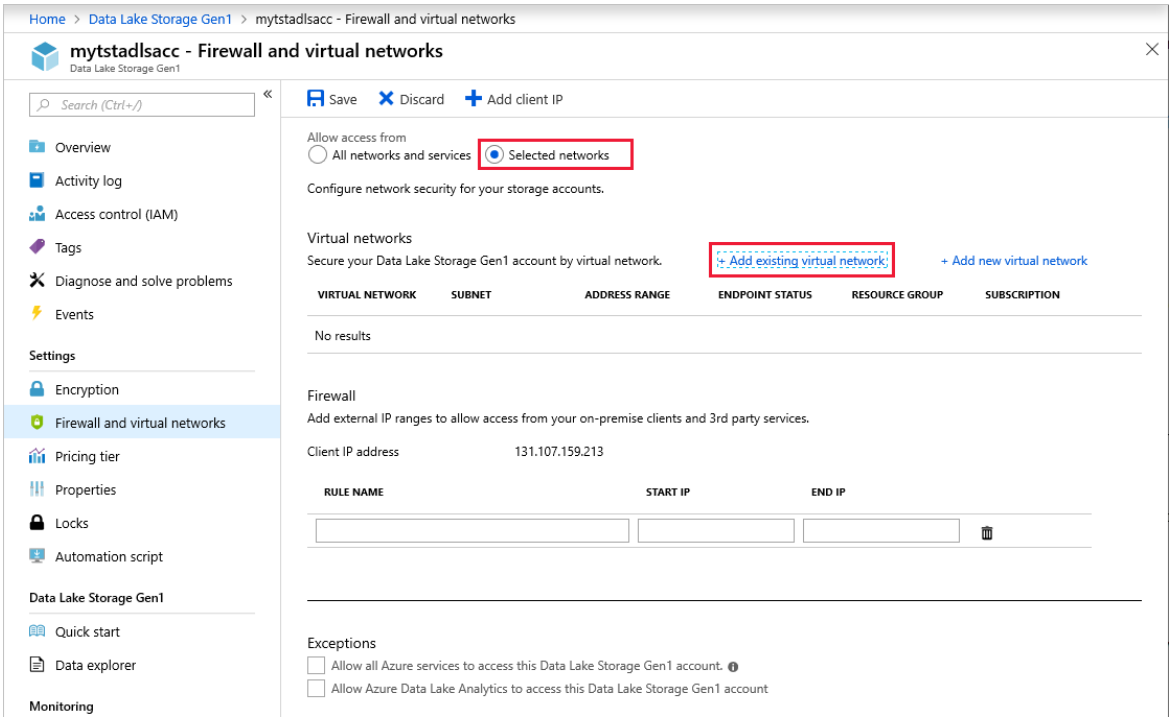
2. Select **Firewall and virtual networks**.

> **NOTE**
>
> If you don't see **Firewall and virtual networks** in the settings, log off the portal. Close the browser, and clear the browser cache. Restart the machine and retry.

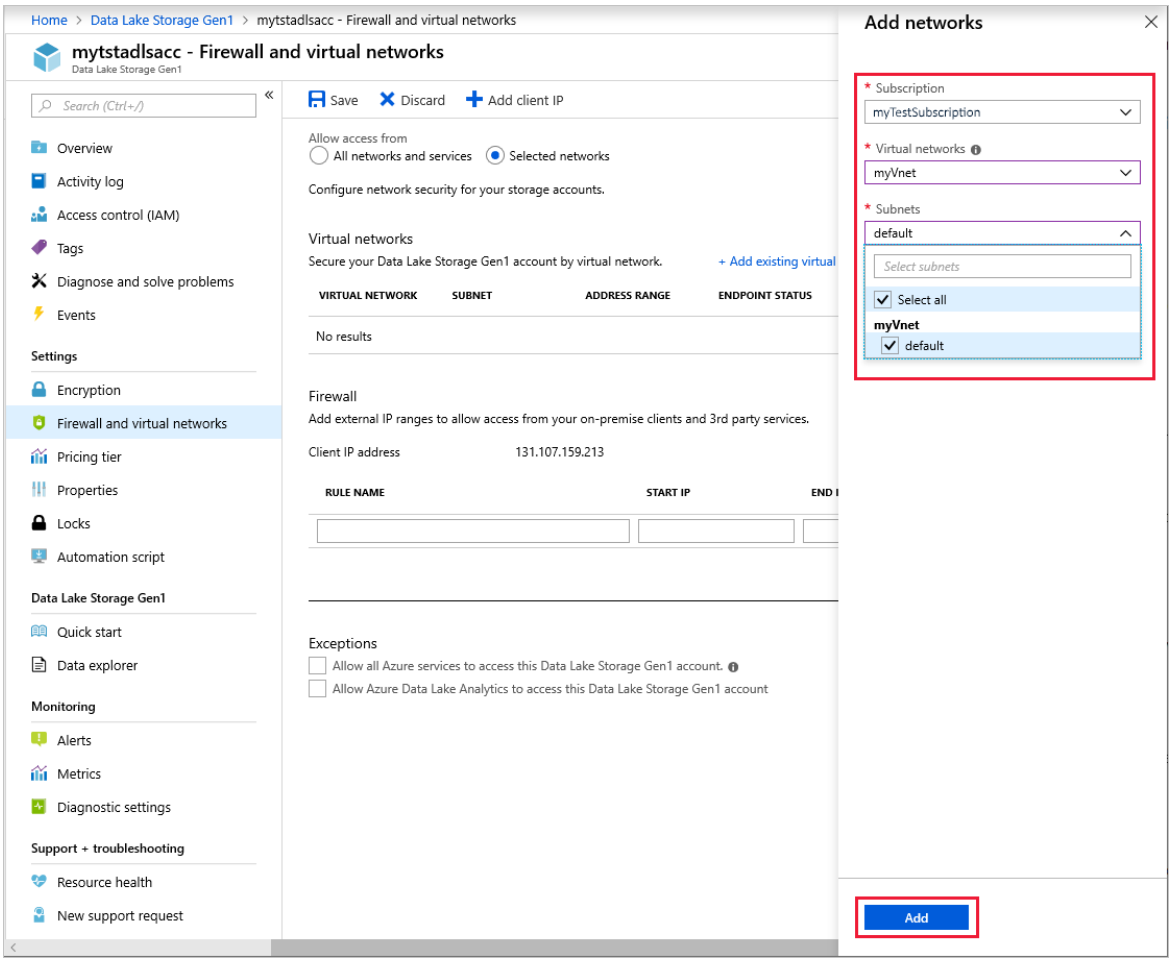


3. Select **Selected networks**.

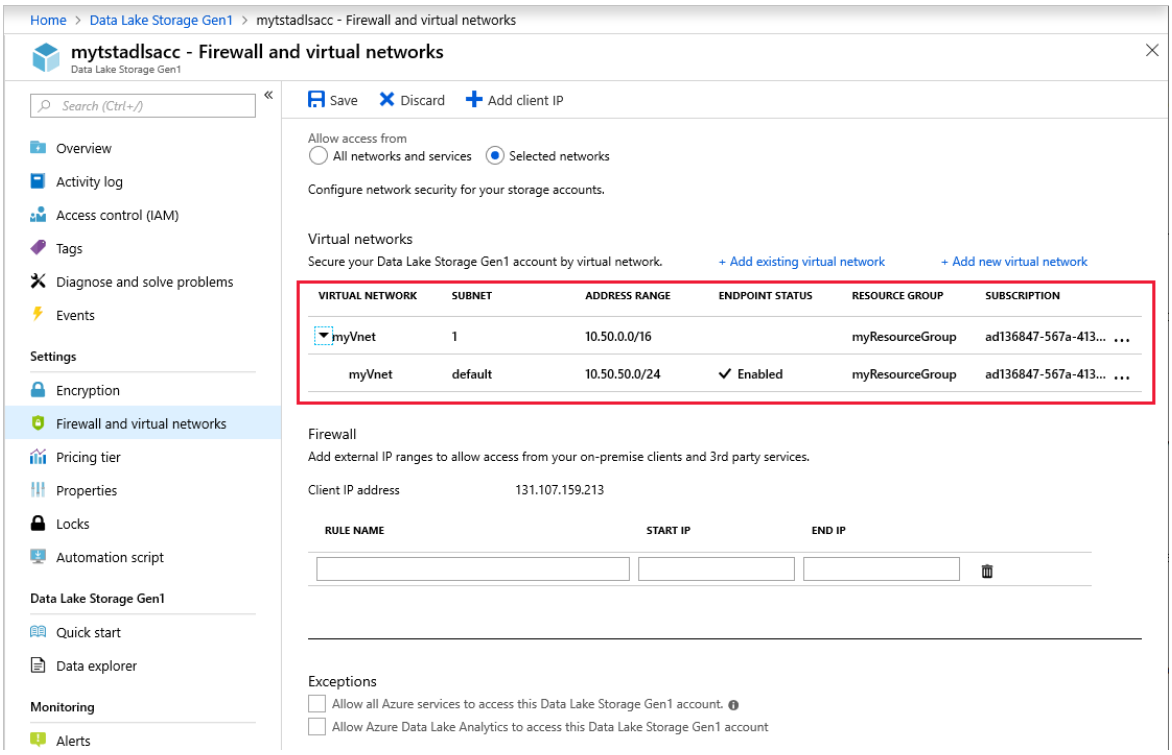
4. Select **Add existing virtual network**.



5. Select the virtual networks and subnets to allow for connectivity. Select **Add**.



6. Make sure that the virtual networks and subnets show up correctly in the list. Select **Save**.



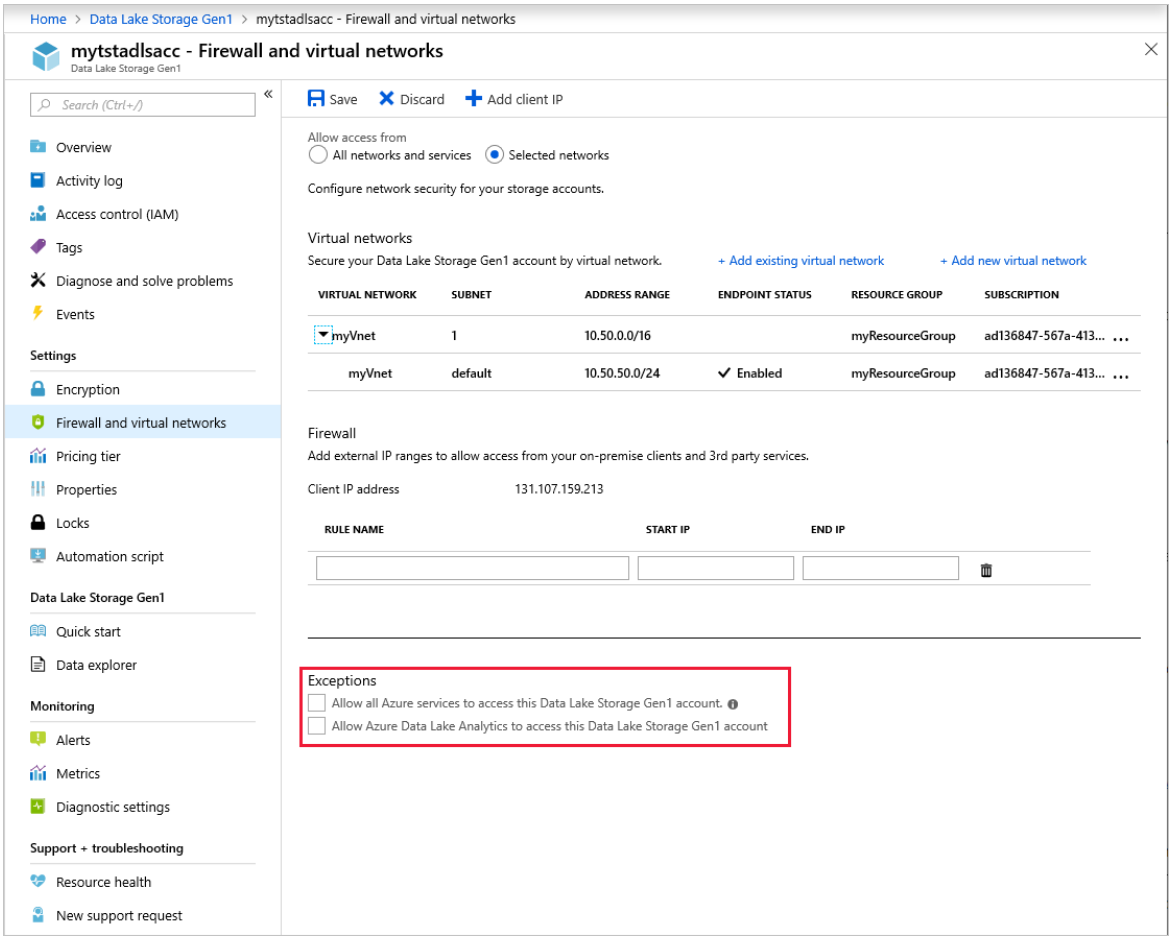
> **NOTE**
>
> It might take up to 5 minutes for the settings to take into effect after you save.

7. [Optional] On the **Firewall and virtual networks** page, in the **Firewall** section, you can allow connectivity from specific IP addresses.

Exceptions

You can enable connectivity from Azure services and VMs outside of your selected virtual networks. On the **Firewall and virtual networks** blade, in the **Exceptions** area, select from two options:

- **Allow all Azure services to access this Data Lake Storage Gen1 account**. This option allows Azure services such as Azure Data Factory, Azure Event Hubs, and all Azure VMs to communicate with your Data Lake Storage account.
- **Allow Azure Data Lake Analytics to access this Data Lake Storage Gen1 account**. This option allows Data Lake Analytics connectivity to this Data Lake Storage account.



We recommend that you keep these exceptions turned off. Turn them on only if you need connectivity from these other services from outside your virtual network.

Authentication with Azure Data Lake Storage Gen1 using Azure Active Directory

10/3/2022 • 2 minutes to read • Edit Online

Azure Data Lake Storage Gen1 uses Azure Active Directory for authentication. Before authoring an application that works with Data Lake Storage Gen1, you must decide how to authenticate your application with Azure Active Directory (Azure AD).

Authentication options

- **End-user authentication** - An end user's Azure credentials are used to authenticate with Data Lake Storage Gen1. The application you create to work with Data Lake Storage Gen1 prompts for these user credentials. As a result, this authentication mechanism is *interactive* and the application runs in the logged in user's context. For more information and instructions, see End-user authentication for Data Lake Storage Gen1.
- **Service-to-service authentication** - Use this option if you want an application to authenticate itself with Data Lake Storage Gen1. In such cases, you create an Azure Active Directory (AD) application and use the key from the Azure AD application to authenticate with Data Lake Storage Gen1. As a result, this authentication mechanism is *non-interactive*. For more information and instructions, see Service-to-service authentication for Data Lake Storage Gen1.

The following table illustrates how end-user and service-to-service authentication mechanisms are supported for Data Lake Storage Gen1. Here's how you read the table.

- The ✔* symbol denotes that authentication option is supported and links to an article that demonstrates how to use the authentication option.
- The ✔ symbol denotes that the authentication option is supported.
- The empty cells denote that the authentication option is not supported.

USE THIS AUTHENTICATION OPTION WITH...	.NET	JAVA	POWERSHELL	AZURE CLI	PYTHON	REST
End-user (without MFA**)	✔	✔	✔	✔	✔*(deprecated)	✔*
End-user (with MFA)	✔*	✔*	✔	✔*	✔*	✔
Service-to-service (using client key)	✔*	✔*	✔	✔	✔*	✔*
Service-to-service (using client certificate)	✔*	✔	✔	✔	✔	✔

** Click the ✔* symbol. It's a link.*

*** MFA stands for multi-factor authentication*

See Authentication Scenarios for Azure Active Directory for more information on how to use Azure Active Directory for authentication.

Next steps

- End-user authentication
- Service-to-service authentication

End-user authentication with Azure Data Lake Storage Gen1 using Azure Active Directory

10/3/2022 • 4 minutes to read • Edit Online

Azure Data Lake Storage Gen1 uses Azure Active Directory for authentication. Before authoring an application that works with Data Lake Storage Gen1 or Azure Data Lake Analytics, you must decide how to authenticate your application with Azure Active Directory (Azure AD). The two main options available are:

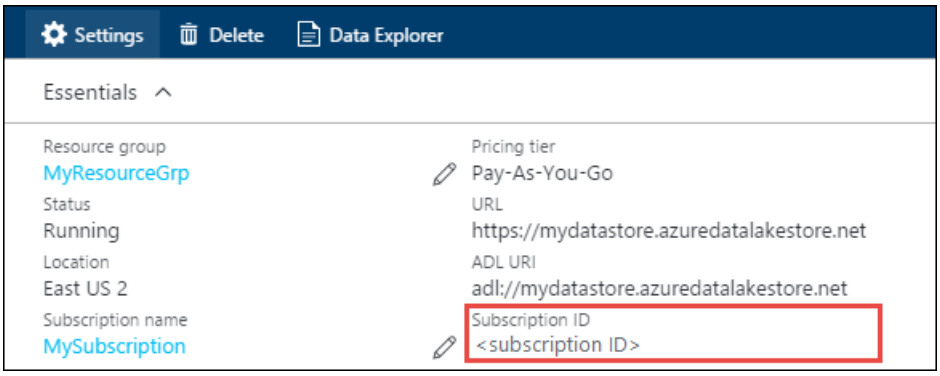
- End-user authentication (this article)
- Service-to-service authentication (pick this option from the drop-down above)

Both these options result in your application being provided with an OAuth 2.0 token, which gets attached to each request made to Data Lake Storage Gen1 or Azure Data Lake Analytics.

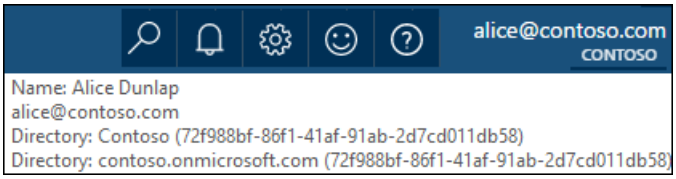
This article talks about how to create an **Azure AD native application for end-user authentication**. For instructions on Azure AD application configuration for service-to-service authentication, see Service-to-service authentication with Data Lake Storage Gen1 using Azure Active Directory.

Prerequisites

- An Azure subscription. See Get Azure free trial.
- Your subscription ID. You can retrieve it from the Azure portal. For example, it is available from the Data Lake Storage Gen1 account blade.



- Your Azure AD domain name. You can retrieve it by hovering the mouse in the top-right corner of the Azure portal. From the screenshot below, the domain name is **contoso.onmicrosoft.com**, and the GUID within brackets is the tenant ID.



- Your Azure tenant ID. For instructions on how to retrieve the tenant ID, see Get the tenant ID.

End-user authentication

This authentication mechanism is the recommended approach if you want an end user to sign in to your application via Azure AD. Your application is then able to access Azure resources with the same level of access as

the end user that logged in. Your end user needs to provide their credentials periodically in order for your application to maintain access.

The result of having the end-user sign in is that your application is given an access token and a refresh token. The access token gets attached to each request made to Data Lake Storage Gen1 or Data Lake Analytics, and it is valid for one hour by default. The refresh token can be used to obtain a new access token, and it is valid for up to two weeks by default. You can use two different approaches for end-user sign in.

Using the OAuth 2.0 pop-up

Your application can trigger an OAuth 2.0 authorization pop-up, in which the end user can enter their credentials. This pop-up also works with the Azure AD Two-factor Authentication (2FA) process, if necessary.

> **NOTE**
> This method is not yet supported in the Azure AD Authentication Library (ADAL) for Python or Java.

Directly passing in user credentials

Your application can directly provide user credentials to Azure AD. This method only works with organizational ID user accounts; it is not compatible with personal / "live ID" user accounts, including the accounts ending in @outlook.com or @live.com. Furthermore, this method is not compatible with user accounts that require Azure AD Two-factor Authentication (2FA).

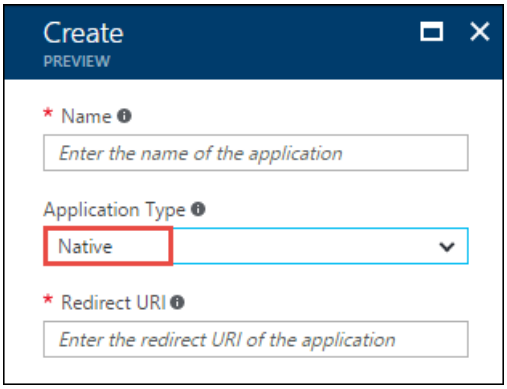
What do I need for this approach?

- Azure AD domain name. This requirement is already listed in the prerequisite of this article.
- Azure AD tenant ID. This requirement is already listed in the prerequisite of this article.
- Azure AD **native application**
- Application ID for the Azure AD native application
- Redirect URI for the Azure AD native application
- Set delegated permissions

Step 1: Create an Active Directory native application

Create and configure an Azure AD native application for end-user authentication with Data Lake Storage Gen1 using Azure Active Directory. For instructions, see Create an Azure AD application.

While following the instructions in the link, make sure you select **Native** for application type, as shown in the following screenshot:

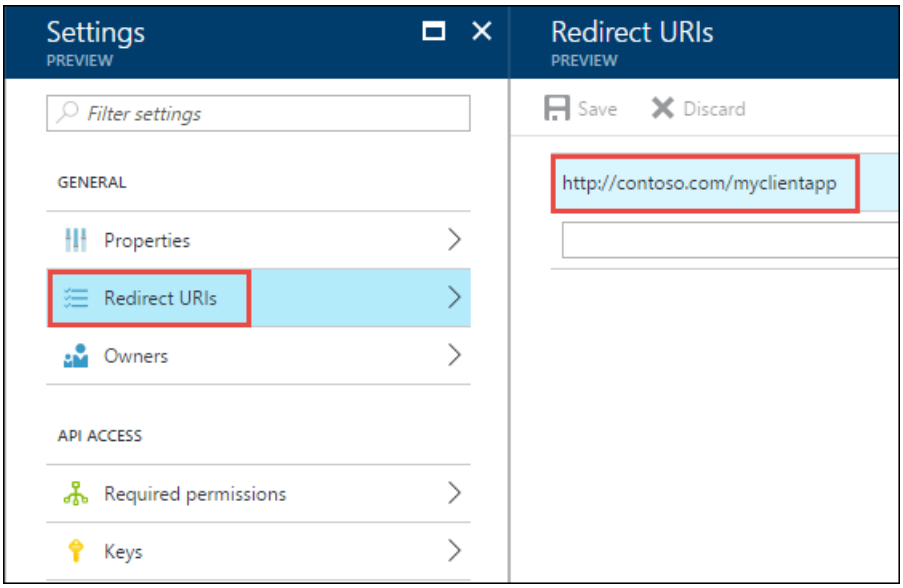


Step 2: Get application ID and redirect URI

See Get the application ID to retrieve the application ID.

To retrieve the redirect URI, do the following steps.

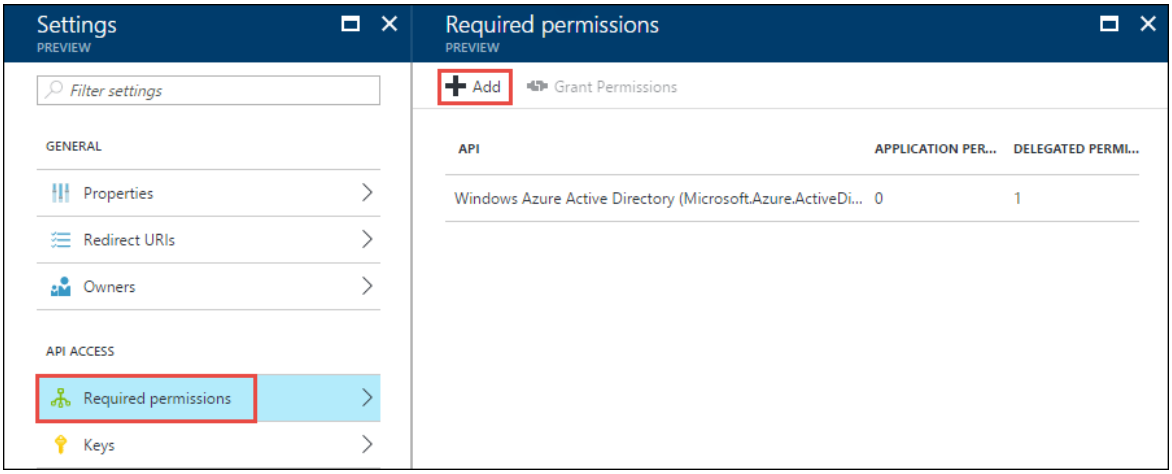
1. From the Azure portal, select **Azure Active Directory**, click **App registrations**, and then find and click the Azure AD native application that you created.
2. From the **Settings** blade for the application, click **Redirect URIs**.



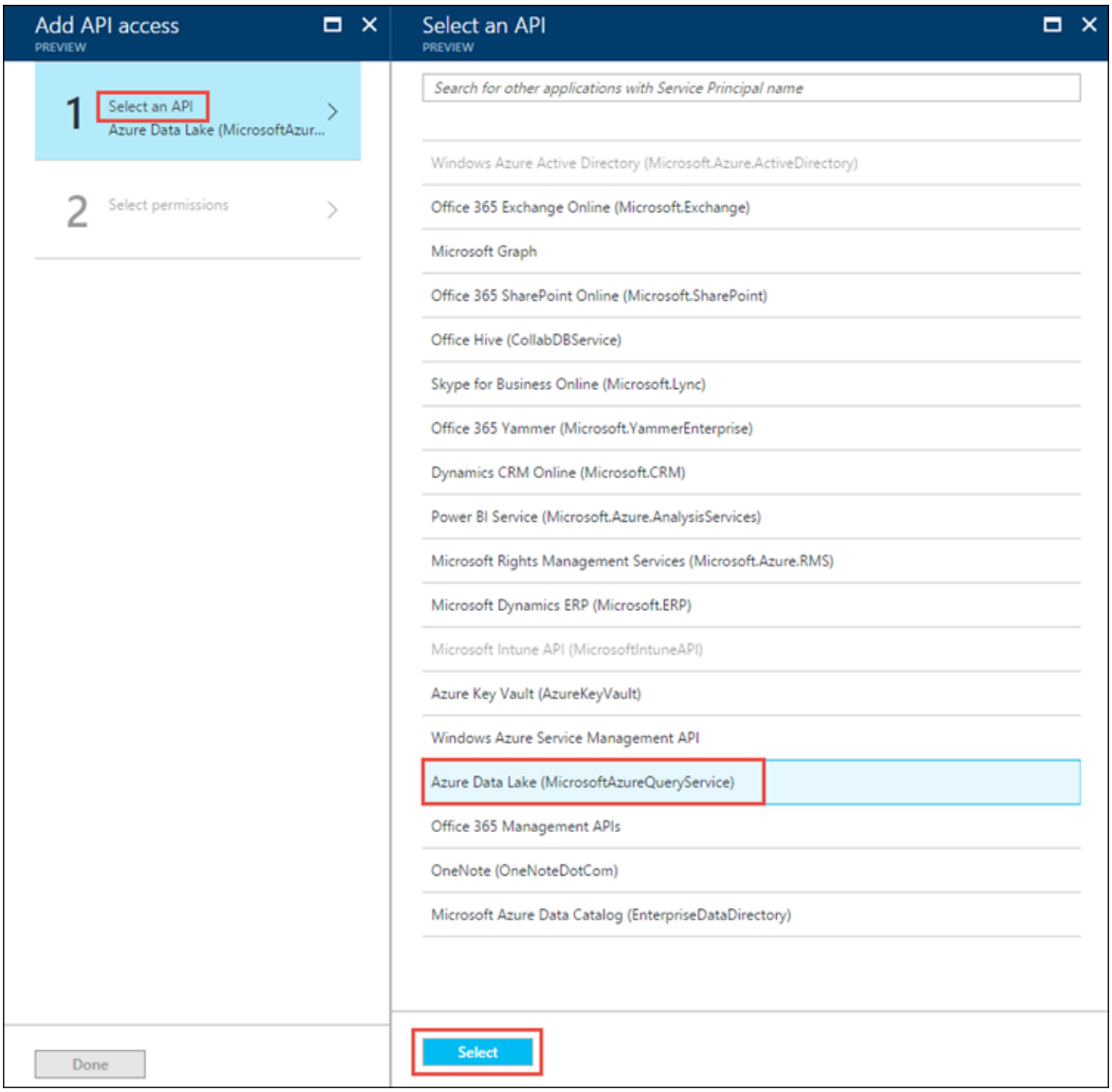
3. Copy the value displayed.

Step 3: Set permissions

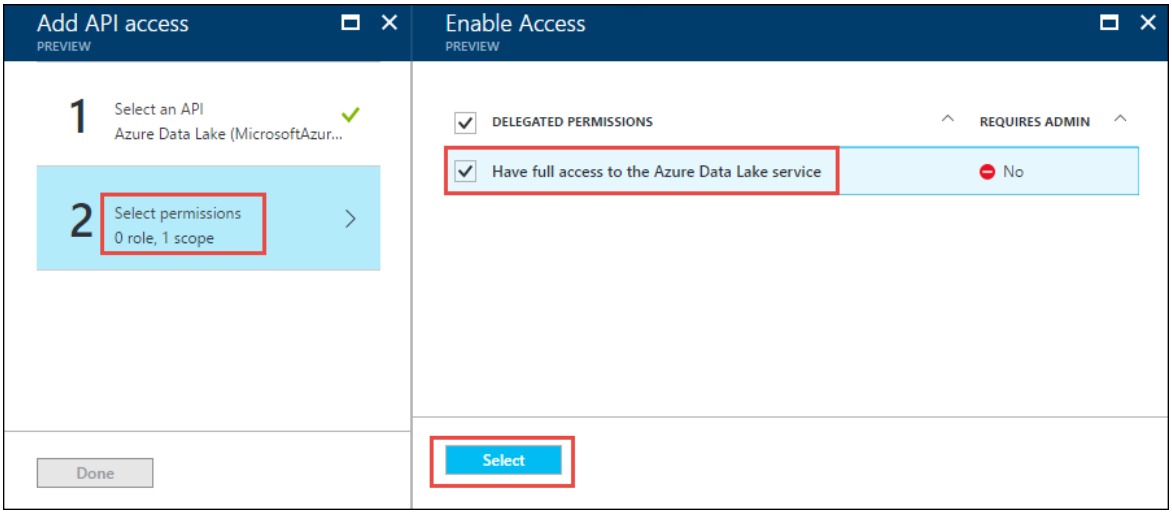
1. From the Azure portal, select **Azure Active Directory**, click **App registrations**, and then find and click the Azure AD native application that you created.
2. From the **Settings** blade for the application, click **Required permissions**, and then click **Add**.



3. In the **Add API Access** blade, click **Select an API**, click **Azure Data Lake**, and then click **Select**.



4. In the **Add API Access** blade, click **Select permissions**, select the check box to give **Full access to Data Lake Store**, and then click **Select**.



Click **Done**.

5. Repeat the last two steps to grant permissions for **Windows Azure Service Management API** as well.

Next steps

In this article, you created an Azure AD native application and gathered the information you need in your client

applications that you author using .NET SDK, Java SDK, REST API, etc. You can now proceed to the following articles that talk about how to use the Azure AD web application to first authenticate with Data Lake Storage Gen1 and then perform other operations on the store.

- End-user-authentication with Data Lake Storage Gen1 using Java SDK
- End-user authentication with Data Lake Storage Gen1 using .NET SDK
- End-user authentication with Data Lake Storage Gen1 using Python
- End-user authentication with Data Lake Storage Gen1 using REST API

End-user authentication with Azure Data Lake Storage Gen1 using Java

10/3/2022 • 2 minutes to read • Edit Online

> **NOTE**
>
> On **Feb 29, 2024** Azure Data Lake Storage Gen1 will be retired. For more information, see the official announcement. If you use Azure Data Lake Storage Gen1, make sure to migrate to Azure Data Lake Storage Gen2 prior to that date. To learn how, see Migrate Azure Data Lake Storage from Gen1 to Gen2 by using the Azure portal.
>
> Unless you already have an Azure Data Lake Storage Gen1 account, you cannot create new ones.

In this article, you learn about how to use the Java SDK to do end-user authentication with Azure Data Lake Storage Gen1. For service-to-service authentication with Data Lake Storage Gen1 using Java SDK, see Service-to-service authentication with Data Lake Storage Gen1 using Java.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Native" Application**. You must have completed the steps in End-user authentication with Data Lake Storage Gen1 using Azure Active Directory.
- Maven. This tutorial uses Maven for build and project dependencies. Although it is possible to build without using a build system like Maven or Gradle, these systems make is much easier to manage dependencies.
- (Optional) And IDE like IntelliJ IDEA or Eclipse or similar.

End-user authentication

1. Create a Maven project using mvn archetype from the command line or using an IDE. For instructions on how to create a Java project using IntelliJ, see here. For instructions on how to create a project using Eclipse, see here.
2. Add the following dependencies to your Maven **pom.xml** file. Add the following snippet before the **</project>** tag:

```
<dependencies>
  <dependency>
    <groupId>com.microsoft.azure</groupId>
    <artifactId>azure-data-lake-store-sdk</artifactId>
    <version>2.2.3</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-nop</artifactId>
    <version>1.7.21</version>
  </dependency>
</dependencies>
```

   The first dependency is to use the Data Lake Storage Gen1 SDK (`azure-data-lake-store-sdk`) from the

maven repository. The second dependency is to specify the logging framework (`slf4j-nop`) to use for this application. The Data Lake Storage Gen1 SDK uses SLF4J logging façade, which lets you choose from a number of popular logging frameworks, like Log4j, Java logging, Logback, etc., or no logging. For this example, we disable logging, hence we use the **slf4j-nop** binding. To use other logging options in your app, see here.

3. Add the following import statements to your application.

```
import com.microsoft.azure.datalake.store.ADLException;
import com.microsoft.azure.datalake.store.ADLStoreClient;
import com.microsoft.azure.datalake.store.DirectoryEntry;
import com.microsoft.azure.datalake.store.IfExists;
import com.microsoft.azure.datalake.store.oauth2.AccessTokenProvider;
import com.microsoft.azure.datalake.store.oauth2.DeviceCodeTokenProvider;
```

4. Use the following snippet in your Java application to obtain token for the Active Directory native application you created earlier using the `DeviceCodeTokenProvider` . Replace **FILL-IN-HERE** with the actual values for the Azure Active Directory native application.

```
private static String nativeAppId = "FILL-IN-HERE";

AccessTokenProvider provider = new DeviceCodeTokenProvider(nativeAppId);
```

The Data Lake Storage Gen1 SDK provides convenient methods that let you manage the security tokens needed to talk to the Data Lake Storage Gen1 account. However, the SDK does not mandate that only these methods be used. You can use any other means of obtaining token as well, like using the Azure Active Directory SDK, or your own custom code.

Next steps

In this article, you learned how to use end-user authentication to authenticate with Azure Data Lake Storage Gen1 using Java SDK. You can now look at the following articles that talk about how to use the Java SDK to work with Azure Data Lake Storage Gen1.

- Data operations on Data Lake Storage Gen1 using Java SDK

End-user authentication with Azure Data Lake Storage Gen1 using .NET SDK

10/3/2022 • 2 minutes to read • Edit Online

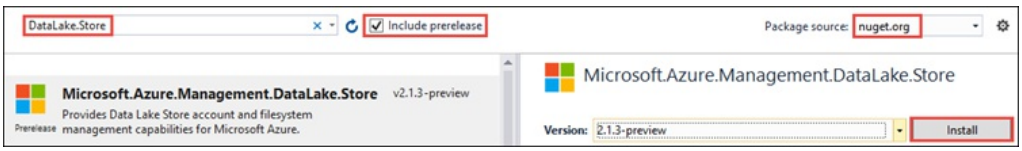
In this article, you learn about how to use the .NET SDK to do end-user authentication with Azure Data Lake Storage Gen1. For service-to-service authentication with Data Lake Storage Gen1 using .NET SDK, see Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK.

Prerequisites

- **Visual Studio 2013 or above**. The instructions below use Visual Studio 2019.
- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Native" Application**. You must have completed the steps in End-user authentication with Data Lake Storage Gen1 using Azure Active Directory.

Create a .NET application

1. In Visual Studio, select the **File** menu, **New**, and then **Project**.
2. Choose **Console App (.NET Framework)**, and then select **Next**.
3. In **Project name**, enter `CreateADLApplication`, and then select **Create**.
4. Add the NuGet packages to your project.
   a. Right-click the project name in the Solution Explorer and click **Manage NuGet Packages**.
   b. In the **NuGet Package Manager** tab, make sure that **Package source** is set to **nuget.org** and that **Include prerelease** check box is selected.
   c. Search for and install the following NuGet packages:
      - `Microsoft.Azure.Management.DataLake.Store` - This tutorial uses v2.1.3-preview.
      - `Microsoft.Rest.ClientRuntime.Azure.Authentication` - This tutorial uses v2.2.12.

   d. Close the **NuGet Package Manager**.
5. Open **Program.cs**
6. Replace the using statements with the following lines:

```
using System;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Collections.Generic;

using Microsoft.Rest;
using Microsoft.Rest.Azure.Authentication;
using Microsoft.Azure.Management.DataLake.Store;
using Microsoft.Azure.Management.DataLake.Store.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
```

End-user authentication

Add this snippet in your .NET client application. Replace the placeholder values with the values retrieved from an Azure AD native application (listed as prerequisite). This snippet lets you authenticate your application **interactively** with Data Lake Storage Gen1, which means you are prompted to enter your Azure credentials.

For ease of use, the following snippet uses default values for client ID and redirect URI that are valid for any Azure subscription. In the following snippet, you only need to provide the value for your tenant ID. You can retrieve the Tenant ID using the instructions provided at Get the tenant ID.

- Replace the Main() function with the following code:

```
private static void Main(string[] args)
{
    //User login via interactive popup
    string TENANT = "<AAD-directory-domain>";
    string CLIENTID = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
    System.Uri ARM_TOKEN_AUDIENCE = new System.Uri(@"https://management.core.windows.net/");
    System.Uri ADL_TOKEN_AUDIENCE = new System.Uri(@"https://datalake.azure.net/");
    string MY_DOCUMENTS =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.MyDocuments);
    string TOKEN_CACHE_PATH = System.IO.Path.Combine(MY_DOCUMENTS, "my.tokencache");
    var tokenCache = GetTokenCache(TOKEN_CACHE_PATH);
    var armCreds = GetCreds_User_Popup(TENANT, ARM_TOKEN_AUDIENCE, CLIENTID, tokenCache);
    var adlCreds = GetCreds_User_Popup(TENANT, ADL_TOKEN_AUDIENCE, CLIENTID, tokenCache);
}
```

A couple of things to know about the preceding snippet:

- The preceding snippet uses a helper functions `GetTokenCache` and `GetCreds_User_Popup`. The code for these helper functions is available here on GitHub.
- To help you complete the tutorial faster, the snippet uses a native application client ID that is available by default for all Azure subscriptions. So, you can **use this snippet as-is in your application**.
- However, if you do want to use your own Azure AD domain and application client ID, you must create an Azure AD native application and then use the Azure AD tenant ID, client ID, and redirect URI for the application you created. See Create an Active Directory Application for end-user authentication with Data Lake Storage Gen1 for instructions.

Next steps

In this article, you learned how to use end-user authentication to authenticate with Azure Data Lake Storage Gen1 using .NET SDK. You can now look at the following articles that talk about how to use the .NET SDK to work with Azure Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using .NET SDK

- Data operations on Data Lake Storage Gen1 using .NET SDK

End-user authentication with Azure Data Lake Storage Gen1 using REST API

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn about how to use the REST API to do end-user authentication with Azure Data Lake Storage Gen1. For service-to-service authentication with Data Lake Storage Gen1 using REST API, see Service-to-service authentication with Data Lake Storage Gen1 using REST API.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Native" Application**. You must have completed the steps in End-user authentication with Data Lake Storage Gen1 using Azure Active Directory.
- **cURL**. This article uses cURL to demonstrate how to make REST API calls against a Data Lake Storage Gen1 account.

End-user authentication

End-user authentication is the recommended approach if you want a user to log in to your application using Azure AD. Your application is able to access Azure resources with the same level of access as the logged-in user. The user needs to provide their credentials periodically in order for your application to maintain access.

The result of having the end-user login is that your application is given an access token and a refresh token. The access token gets attached to each request made to Data Lake Storage Gen1 or Data Lake Analytics, and it is valid for one hour by default. The refresh token can be used to obtain a new access token, and it is valid for up to two weeks by default, if used regularly. You can use two different approaches for end-user login.

In this scenario, the application prompts the user to log in and all the operations are performed in the context of the user. Perform the following steps:

1. Through your application, redirect the user to the following URL:

   ```
   https://login.microsoftonline.com/<TENANT-ID>/oauth2/authorize?client_id=<APPLICATION-ID>&response_type=code&redirect_uri=<REDIRECT-URI>
   ```

   > **NOTE**
   > <REDIRECT-URI> needs to be encoded for use in a URL. So, for https://localhost, use `https%3A%2F%2Flocalhost`
   >)

   For the purpose of this tutorial, you can replace the placeholder values in the URL above and paste it in a web browser's address bar. You will be redirected to authenticate using your Azure login. Once you successfully log in, the response is displayed in the browser's address bar. The response will be in the following format:

   ```
   http://localhost/?code=<AUTHORIZATION-CODE>&session_state=<GUID>
   ```

2. Capture the authorization code from the response. For this tutorial, you can copy the authorization code from the address bar of the web browser and pass it in the POST request to the token endpoint, as shown in the following snippet:

```
curl -X POST https://login.microsoftonline.com/<TENANT-ID>/oauth2/token \
-F redirect_uri=<REDIRECT-URI> \
-F grant_type=authorization_code \
-F resource=https://management.core.windows.net/ \
-F client_id=<APPLICATION-ID> \
-F code=<AUTHORIZATION-CODE>
```

> **NOTE**
>
> In this case, the <REDIRECT-URI> need not be encoded.

3. The response is a JSON object that contains an access token (for example, `"access_token": "<ACCESS_TOKEN>"`) and a refresh token (for example, `"refresh_token": "<REFRESH_TOKEN>"`). Your application uses the access token when accessing Azure Data Lake Storage Gen1 and the refresh token to get another access token when an access token expires.

```
{"token_type":"Bearer","scope":"user_impersonation","expires_in":"3599","expires_on":"1461865782","no
t_before":    "1461861882","resource":"https://management.core.windows.net/","access_token":"
<REDACTED>","refresh_token":"<REDACTED>","id_token":"<REDACTED>"}
```

4. When the access token expires, you can request a new access token using the refresh token, as shown in the following snippet:

```
curl -X POST https://login.microsoftonline.com/<TENANT-ID>/oauth2/token  \
     -F grant_type=refresh_token \
     -F resource=https://management.core.windows.net/ \
     -F client_id=<APPLICATION-ID> \
     -F refresh_token=<REFRESH-TOKEN>
```

For more information on interactive user authentication, see Authorization code grant flow.

Next steps

In this article, you learned how to use service-to-service authentication to authenticate with Azure Data Lake Storage Gen1 using REST API. You can now look at the following articles that talk about how to use the REST API to work with Azure Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using REST API
- Data operations on Data Lake Storage Gen1 using REST API

End-user authentication with Azure Data Lake Storage Gen1 using Python

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn about how to use the Python SDK to do end-user authentication with Azure Data Lake Storage Gen1. End-user authentication can further be split into two categories:

- End-user authentication without multi-factor authentication
- End-user authentication with multi-factor authentication

Both these options are discussed in this article. For service-to-service authentication with Data Lake Storage Gen1 using Python, see Service-to-service authentication with Data Lake Storage Gen1 using Python.

Prerequisites

- **Python**. You can download Python from here. This article uses Python 3.6.2.
- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Native" Application**. You must have completed the steps in End-user authentication with Data Lake Storage Gen1 using Azure Active Directory.

Install the modules

To work with Data Lake Storage Gen1 using Python, you need to install three modules.

- The `azure-mgmt-resource` module, which includes Azure modules for Active Directory, etc.
- The `azure-mgmt-datalake-store` module, which includes the Azure Data Lake Storage Gen1 account management operations. For more information on this module, see Azure Data Lake Storage Gen1 Management module reference.
- The `azure-datalake-store` module, which includes the Azure Data Lake Storage Gen1 filesystem operations. For more information on this module, see azure-datalake-store Filesystem module reference.

Use the following commands to install the modules.

```
pip install azure-mgmt-resource
pip install azure-mgmt-datalake-store
pip install azure-datalake-store
```

Create a new Python application

1. In the IDE of your choice, create a new Python application, for example, `mysample.py`.
2. Add the following snippet to import the required modules

```
## Use this for Azure AD authentication
from msrestazure.azure_active_directory import AADTokenCredentials

## Required for Azure Data Lake Storage Gen1 account management
from azure.mgmt.datalake.store import DataLakeStoreAccountManagementClient
from azure.mgmt.datalake.store.models import DataLakeStoreAccount

## Required for Azure Data Lake Storage Gen1 filesystem management
from azure.datalake.store import core, lib, multithread

 # Common Azure imports
 import adal
 from azure.mgmt.resource.resources import ResourceManagementClient
 from azure.mgmt.resource.resources.models import ResourceGroup

 ## Use these as needed for your application
 import logging, pprint, uuid, time
```

3. Save changes to `mysample.py`.

End-user authentication with multi-factor authentication

For account management

Use the following snippet to authenticate with Azure AD for account management operations on a Data Lake Storage Gen1 account. The following snippet can be used to authenticate your application using multi-factor authentication. Provide the values below for an existing Azure AD **native** application.

```
authority_host_url = "https://login.microsoftonline.com"
tenant = "FILL-IN-HERE"
authority_url = authority_host_url + '/' + tenant
client_id = 'FILL-IN-HERE'
redirect = 'urn:ietf:wg:oauth:2.0:oob'
RESOURCE = 'https://management.core.windows.net/'

context = adal.AuthenticationContext(authority_url)
code = context.acquire_user_code(RESOURCE, client_id)
print(code['message'])
mgmt_token = context.acquire_token_with_device_code(RESOURCE, code, client_id)
armCreds = AADTokenCredentials(mgmt_token, client_id, resource = RESOURCE)
```

For filesystem operations

Use this to authenticate with Azure AD for filesystem operations on a Data Lake Storage Gen1 account. The following snippet can be used to authenticate your application using multi-factor authentication. Provide the values below for an existing Azure AD **native** application.

```
adlCreds = lib.auth(tenant_id='FILL-IN-HERE', resource = 'https://datalake.azure.net/')
```

End-user authentication without multi-factor authentication

This is deprecated. For more information, see Azure Authentication using Python SDK.

Next steps

In this article, you learned how to use end-user authentication to authenticate with Azure Data Lake Storage Gen1 using Python. You can now look at the following articles that talk about how to use Python to work with Azure Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using Python
- Data operations on Data Lake Storage Gen1 using Python

Service-to-service authentication with Azure Data Lake Storage Gen1 using Azure Active Directory

10/3/2022 • 3 minutes to read • Edit Online

Azure Data Lake Storage Gen1 uses Azure Active Directory for authentication. Before authoring an application that works with Data Lake Storage Gen1, you must decide how to authenticate your application with Azure Active Directory (Azure AD). The two main options available are:

- End-user authentication
- Service-to-service authentication (this article)

Both these options result in your application being provided with an OAuth 2.0 token, which gets attached to each request made to Data Lake Storage Gen1.

This article talks about how to create an **Azure AD web application for service-to-service authentication**. For instructions on Azure AD application configuration for end-user authentication, see End-user authentication with Data Lake Storage Gen1 using Azure Active Directory.

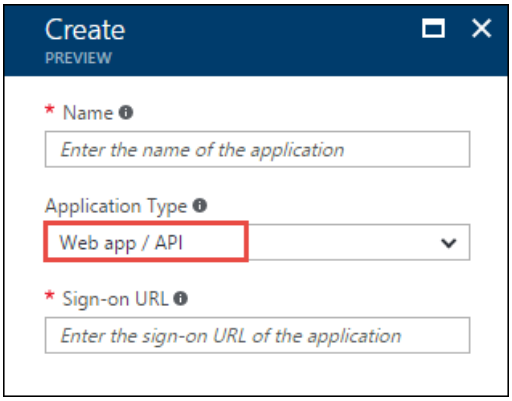
Prerequisites

- An Azure subscription. See Get Azure free trial.

Step 1: Create an Active Directory web application

Create and configure an Azure AD web application for service-to-service authentication with Azure Data Lake Storage Gen1 using Azure Active Directory. For instructions, see Create an Azure AD application.

While following the instructions at the preceding link, make sure you select **Web App / API** for application type, as shown in the following screenshot:



Step 2: Get application ID, authentication key, and tenant ID

When programmatically logging in, you need the ID for your application. If the application runs under its own credentials, you also need an authentication key.

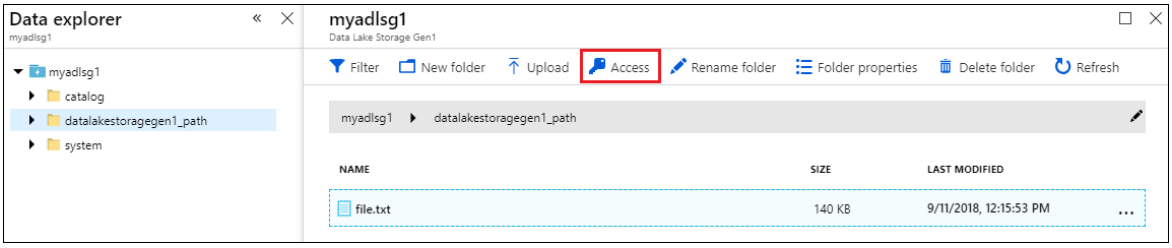
- For instructions on how to retrieve the application ID and authentication key (also called the client secret) for your application, see Get application ID and authentication key.
- For instructions on how to retrieve the tenant ID, see Get tenant ID.

Step 3: Assign the Azure AD application to the Azure Data Lake Storage Gen1 account file or folder

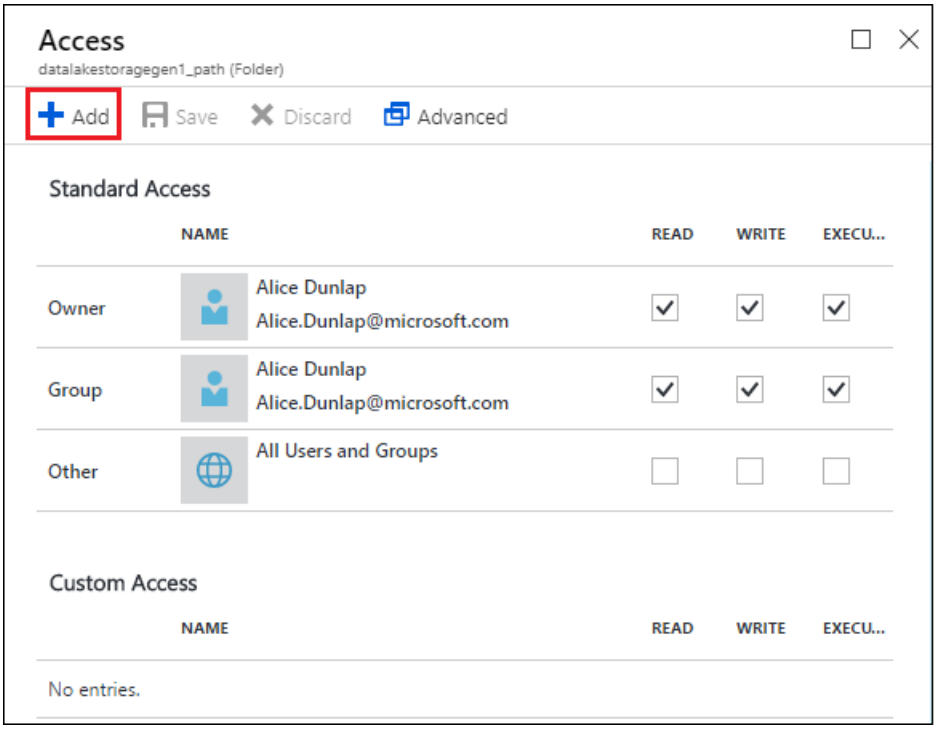
1. Sign on to the Azure portal. Open the Data Lake Storage Gen1 account that you want to associate with the Azure Active Directory application you created earlier.

2. In your Data Lake Storage Gen1 account blade, click **Data Explorer**.

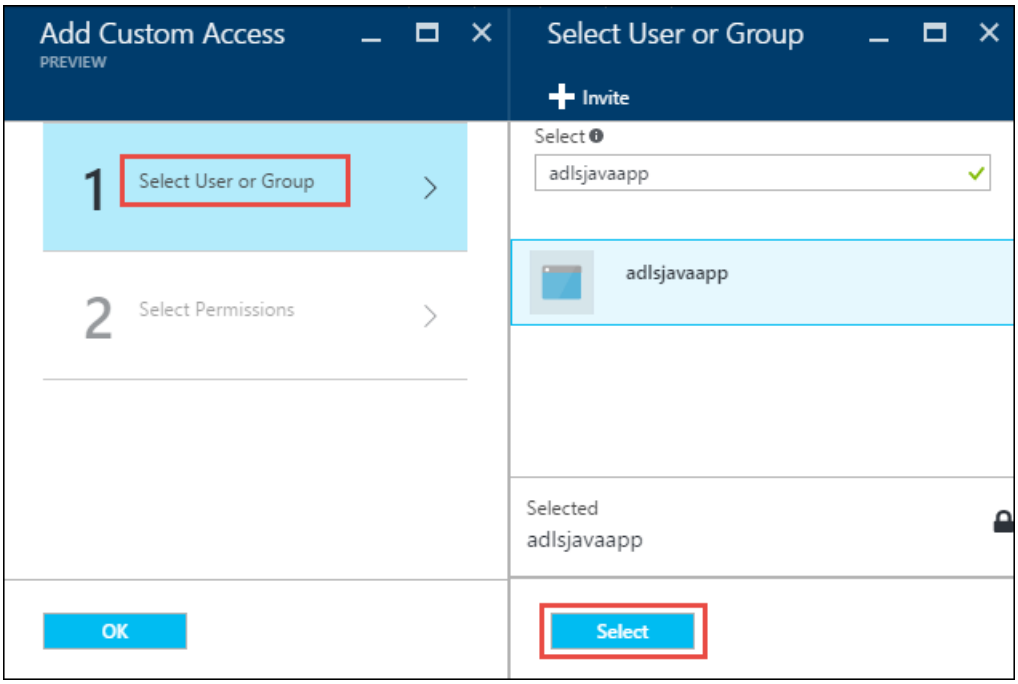
3. In the **Data Explorer** blade, click the file or folder for which you want to provide access to the Azure AD application, and then click **Access**. To configure access to a file, you must click **Access** from the **File Preview** blade.



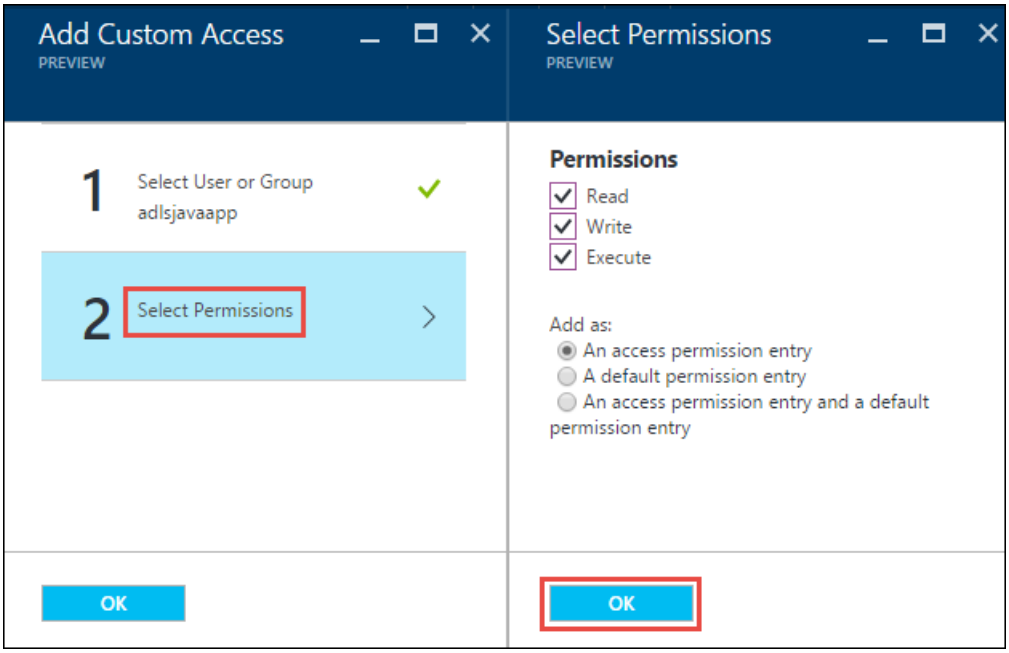
4. The **Access** blade lists the standard access and custom access already assigned to the root. Click the **Add** icon to add custom-level ACLs.



5. Click the **Add** icon to open the **Add Custom Access** blade. In this blade, click **Select User or Group**, and then in **Select User or Group** blade, look for the Azure Active Directory application you created earlier. If you have many groups to search from, use the text box at the top to filter on the group name. Click the group you want to add and then click **Select**.

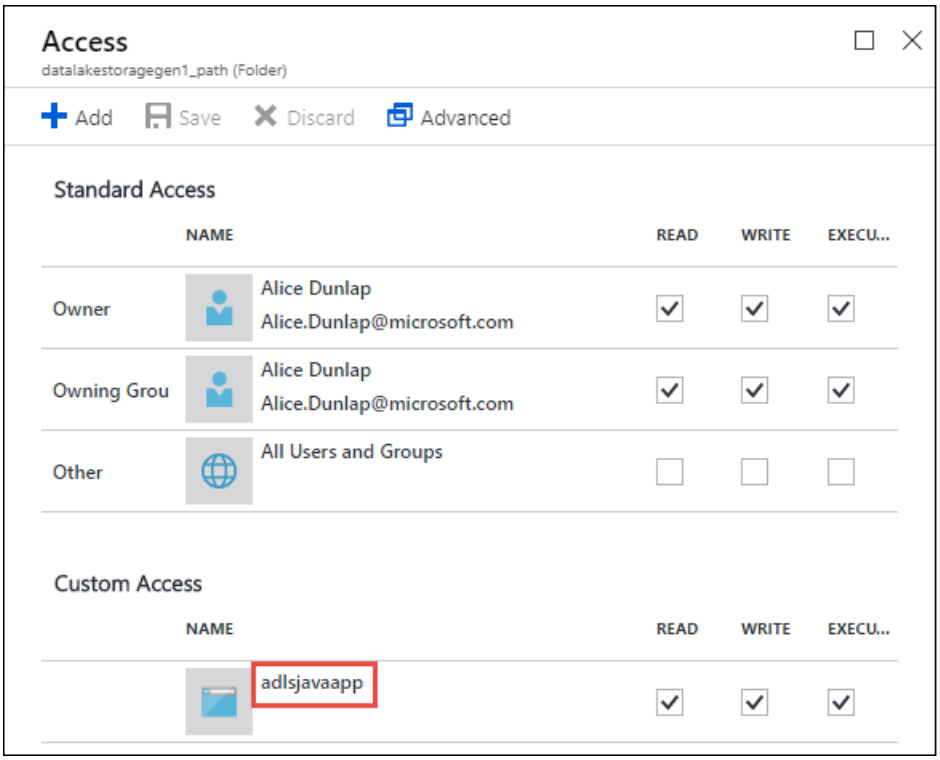


6. Click **Select Permissions**, select the permissions and whether you want to assign the permissions as a default ACL, access ACL, or both. Click **OK**.



For more information about permissions in Data Lake Storage Gen1, and Default/Access ACLs, see Access Control in Data Lake Storage Gen1.

7. In the **Add Custom Access** blade, click **OK**. The newly added groups, with the associated permissions, are listed in the **Access** blade.



NOTE

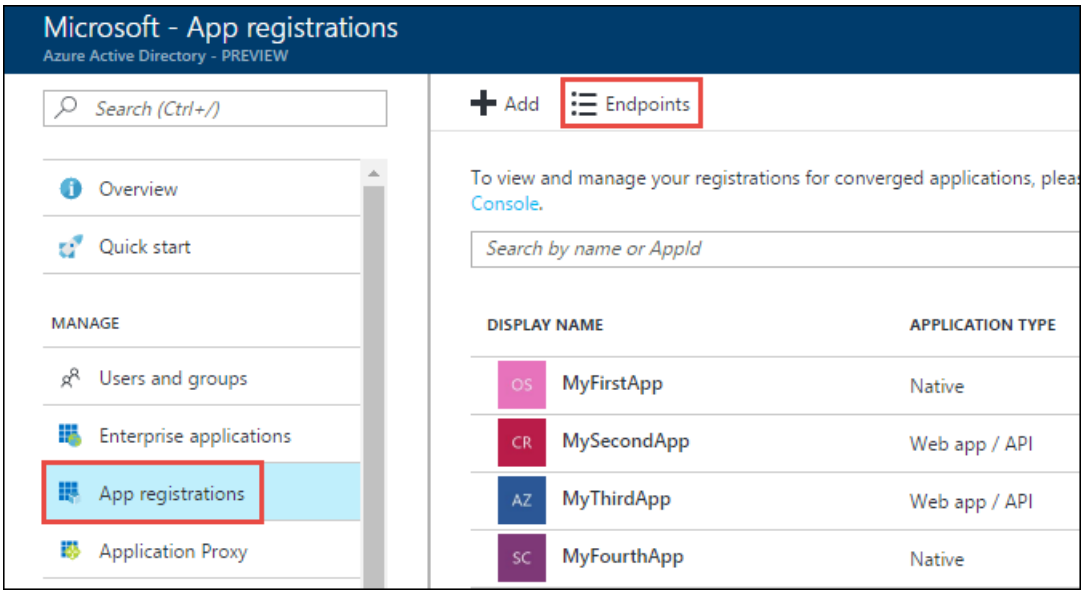
If you plan on restricting your Azure Active Directory application to a specific folder, you will also need to give that same Azure Active directory application **Execute** permission to the root to enable file creation access via the .NET SDK.

NOTE

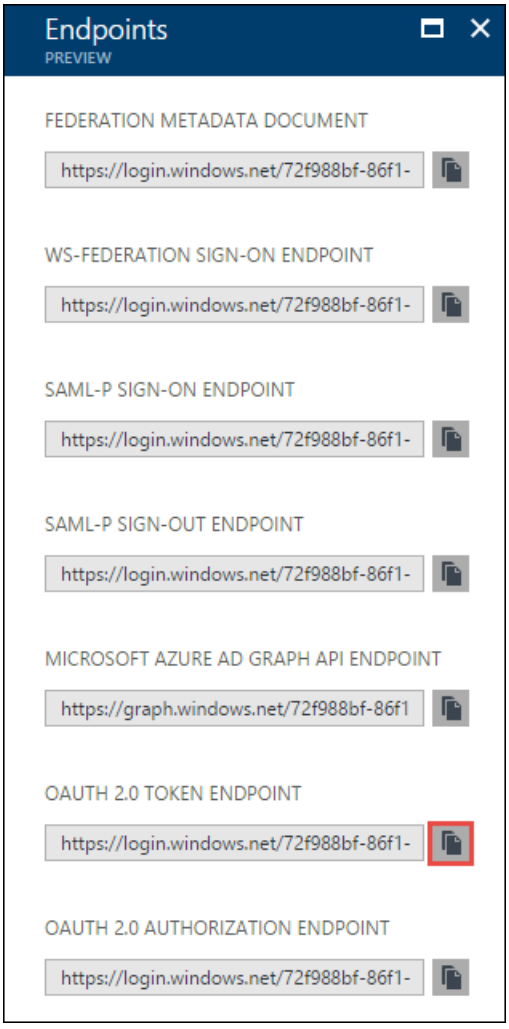
If you want to use the SDKs to create a Data Lake Storage Gen1 account, you must assign the Azure AD web application as a role to the Resource Group in which you create the Data Lake Storage Gen1 account.

Step 4: Get the OAuth 2.0 token endpoint (only for Java-based applications)

1. Sign on to the Azure portal and click Active Directory from the left pane.
2. From the left pane, click **App registrations**.
3. From the top of the App registrations blade, click **Endpoints**.



4. From the list of endpoints, copy the OAuth 2.0 token endpoint.



Next steps

In this article, you created an Azure AD web application and gathered the information you need in your client applications that you author using .NET SDK, Java, Python, REST API, etc. You can now proceed to the following articles that talk about how to use the Azure AD native application to first authenticate with Data Lake Storage Gen1 and then perform other operations on the store.

- Service-to-service authentication with Data Lake Storage Gen1 using Java
- Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK

- Service-to-service authentication with Data Lake Storage Gen1 using Python
- Service-to-service authentication with Data Lake Storage Gen1 using REST API

Service-to-service authentication with Azure Data Lake Storage Gen2 using Java

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn about how to use the Java SDK to do service-to-service authentication with Azure Data Lake Storage Gen2. End-user authentication with Data Lake Storage Gen2 using the Java SDK isn't supported.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Web" Application**. You must have completed the steps in Service-to-service authentication with Data Lake Storage Gen2 using Azure Active Directory.
- Maven. This tutorial uses Maven for build and project dependencies. Although it is possible to build without using a build system like Maven or Gradle, these systems make it much easier to manage dependencies.
- (Optional) An IDE like IntelliJ IDEA or Eclipse or similar.

Service-to-service authentication

1. Create a Maven project using mvn archetype from the command line or using an IDE. For instructions on how to create a Java project using IntelliJ, see here. For instructions on how to create a project using Eclipse, see here.
2. Add the following dependencies to your Maven **pom.xml** file. Add the following snippet before the **</project>** tag:

```
<dependencies>
  <dependency>
      <groupId>com.azure</groupId>
      <artifactId>azure-storage-file-datalake</artifactId>
      <version>12.6.0</version>
  </dependency>
  <dependency>
      <groupId>com.azure</groupId>
      <artifactId>azure-identity</artifactId>
      <version>1.3.3</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-nop</artifactId>
    <version>1.7.21</version>
  </dependency>
</dependencies>
```

   The first dependency is to use the Data Lake Storage Gen2 SDK (`azure-storage-file-datalake`) from the Maven repository. The second dependency is to specify the logging framework (`slf4j-nop`) to use for this app. The Data Lake Storage Gen2 SDK uses the slf4j logging façade, which lets you choose from a number of popular logging frameworks, like log4j, Java logging, logback, or no logging. For this example, we disable logging, hence we use the **slf4j-nop** binding. To use other logging options in your app, see Declaring project dependencies for logging.

3. Add the following import statements to your application.

```
import com.azure.identity.ClientSecretCredential;
import com.azure.identity.ClientSecretCredentialBuilder;
import com.azure.storage.file.datalake.DataLakeDirectoryClient;
import com.azure.storage.file.datalake.DataLakeFileClient;
import com.azure.storage.file.datalake.DataLakeServiceClient;
import com.azure.storage.file.datalake.DataLakeServiceClientBuilder;
import com.azure.storage.file.datalake.DataLakeFileSystemClient;
import com.azure.storage.file.datalake.models.ListPathsOptions;
import com.azure.storage.file.datalake.models.PathAccessControl;
import com.azure.storage.file.datalake.models.PathPermissions;
```

4. Use the following snippet in your Java app to obtain a token for the Active Directory web app you created earlier using one of the class of `StorageSharedKeyCredential` (the following example uses `credential`). The token provider caches the credentials used to obtain the token in memory, and automatically renews the token if it's about to expire. It's possible to create your own subclasses of `StorageSharedKeyCredential` so tokens are obtained by your customer code. For now, let's just use the one provided in the SDK.

   Replace **FILL-IN-HERE** with the actual values for the Azure Active Directory Web application.

```
private static String clientId = "FILL-IN-HERE";
private static String tenantId = "FILL-IN-HERE";
private static String clientSecret = "FILL-IN-HERE";

ClientSecretCredential credential = new
ClientSecretCredentialBuilder().clientId(clientId).tenantId(tenantId).clientSecret(clientSecret).buil
d();
```

The Data Lake Storage Gen2 SDK provides convenient methods that let you manage the security tokens needed to talk to the Data Lake Storage Gen2 account. However, the SDK doesn't mandate that only these methods be used. You can use any other means of obtaining token as well, like using the Azure Identity client library or your own custom code.

Next steps

In this article, you learned how to use end-user authentication to authenticate with Data Lake Storage Gen2 using Java SDK. You can now look at the following articles that talk about how to use the Java SDK to work with Data Lake Storage Gen2.

- Data operations on Data Lake Storage Gen2 using Java SDK

Service-to-service authentication with Azure Data Lake Storage Gen1 using .NET SDK

10/3/2022 • 3 minutes to read • Edit Online

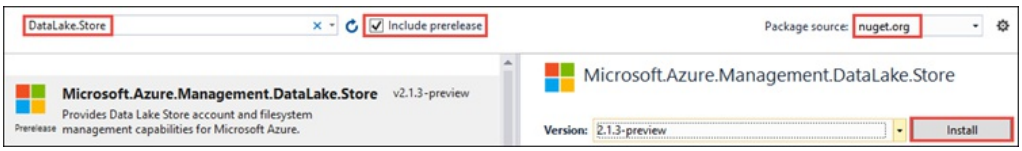
In this article, you learn about how to use the .NET SDK to do service-to-service authentication with Azure Data Lake Storage Gen1. For end-user authentication with Data Lake Storage Gen1 using .NET SDK, see End-user authentication with Data Lake Storage Gen1 using .NET SDK.

Prerequisites

- **Visual Studio 2013 or above**. The instructions below use Visual Studio 2019.
- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Web" Application**. You must have completed the steps in Service-to-service authentication with Data Lake Storage Gen1 using Azure Active Directory.

Create a .NET application

1. In Visual Studio, select the **File** menu, **New**, and then **Project**.
2. Choose **Console App (.NET Framework)**, and then select **Next**.
3. In **Project name**, enter `CreateADLApplication`, and then select **Create**.
4. Add the NuGet packages to your project.
   a. Right-click the project name in the Solution Explorer and click **Manage NuGet Packages**.
   b. In the **NuGet Package Manager** tab, make sure that **Package source** is set to **nuget.org** and that **Include prerelease** check box is selected.
   c. Search for and install the following NuGet packages:
      - `Microsoft.Azure.Management.DataLake.Store` - This tutorial uses v2.1.3-preview.
      - `Microsoft.Rest.ClientRuntime.Azure.Authentication` - This tutorial uses v2.2.12.

   d. Close the **NuGet Package Manager**.
5. Open **Program.cs**, delete the existing code, and then include the following statements to add references to namespaces.

```
using System;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Collections.Generic;
using System.Security.Cryptography.X509Certificates; // Required only if you are using an Azure AD
application created with certificates

using Microsoft.Rest;
using Microsoft.Rest.Azure.Authentication;
using Microsoft.Azure.Management.DataLake.Store;
using Microsoft.Azure.Management.DataLake.Store.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
```

Service-to-service authentication with client secret

Add this snippet in your .NET client application. Replace the placeholder values with the values retrieved from an Azure AD web application (listed as a prerequisite). This snippet lets you authenticate your application **non-interactively** with Data Lake Storage Gen1 using the client secret/key for Azure AD web application.

```
private static void Main(string[] args)
{
    // Service principal / application authentication with client secret / key
    // Use the client ID of an existing AAD "Web App" application.
    string TENANT = "<AAD-directory-domain>";
    string CLIENTID = "<AAD_WEB_APP_CLIENT_ID>";
    System.Uri ARM_TOKEN_AUDIENCE = new System.Uri(@"https://management.core.windows.net/");
    System.Uri ADL_TOKEN_AUDIENCE = new System.Uri(@"https://datalake.azure.net/");
    string secret_key = "<AAD_WEB_APP_SECRET_KEY>";
    var armCreds = GetCreds_SPI_SecretKey(TENANT, ARM_TOKEN_AUDIENCE, CLIENTID, secret_key);
    var adlCreds = GetCreds_SPI_SecretKey(TENANT, ADL_TOKEN_AUDIENCE, CLIENTID, secret_key);
}
```

The preceding snippet uses a helper function `GetCreds_SPI_SecretKey`. The code for this helper function is available here on GitHub.

Service-to-service authentication with certificate

Add this snippet in your .NET client application. Replace the placeholder values with the values retrieved from an Azure AD web application (listed as a prerequisite). This snippet lets you authenticate your application **non-interactively** with Data Lake Storage Gen1 using the certificate for an Azure AD web application. For instructions on how to create an Azure AD application, see Create service principal with certificates.

```
private static void Main(string[] args)
{
    // Service principal / application authentication with certificate
    // Use the client ID and certificate of an existing AAD "Web App" application.
    string TENANT = "<AAD-directory-domain>";
    string CLIENTID = "<AAD_WEB_APP_CLIENT_ID>";
    System.Uri ARM_TOKEN_AUDIENCE = new System.Uri(@"https://management.core.windows.net/");
    System.Uri ADL_TOKEN_AUDIENCE = new System.Uri(@"https://datalake.azure.net/");
    var cert = new X509Certificate2(@"d:\cert.pfx", "<certpassword>");
    var armCreds = GetCreds_SPI_Cert(TENANT, ARM_TOKEN_AUDIENCE, CLIENTID, cert);
    var adlCreds = GetCreds_SPI_Cert(TENANT, ADL_TOKEN_AUDIENCE, CLIENTID, cert);
}
```

The preceding snippet uses a helper function `GetCreds_SPI_Cert`. The code for this helper function is available

here on GitHub.

Next steps

In this article, you learned how to use service-to-service authentication to authenticate with Data Lake Storage Gen1 using .NET SDK. You can now look at the following articles that talk about how to use the .NET SDK to work with Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using .NET SDK
- Data operations on Data Lake Storage Gen1 using .NET SDK

Service-to-service authentication with Azure Data Lake Storage Gen1 using REST API

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn how to use the REST API to do service-to-service authentication with Azure Data Lake Storage Gen1. For end-user authentication with Data Lake Storage Gen1 using REST API, see End-user authentication with Data Lake Storage Gen1 using REST API.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Web" Application**. You must have completed the steps in Service-to-service authentication with Data Lake Storage Gen1 using Azure Active Directory.

Service-to-service authentication

In this scenario, the application provides its own credentials to perform the operations. For this, you must issue a POST request like the one shown in the following snippet:

```
curl -X POST https://login.microsoftonline.com/<TENANT-ID>/oauth2/token  \
  -F grant_type=client_credentials \
  -F resource=https://management.core.windows.net/ \
  -F client_id=<CLIENT-ID> \
  -F client_secret=<AUTH-KEY>
```

The output of the request includes an authorization token (denoted by `access-token` in the output below) that you subsequently pass with your REST API calls. Save the authentication token in a text file; you will need it when making REST calls to Data Lake Storage Gen1.

```
{"token_type":"Bearer","expires_in":"3599","expires_on":"1458245447","not_before":"1458241547","resource":"h
ttps://management.core.windows.net/","access_token":"<REDACTED>"}
```

This article uses the **non-interactive** approach. For more information on non-interactive (service-to-service calls), see Service to service calls using credentials.

Next steps

In this article, you learned how to use service-to-service authentication to authenticate with Data Lake Storage Gen1 using REST API. You can now look at the following articles that talk about how to use the REST API to work with Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using REST API
- Data operations on Data Lake Storage Gen1 using REST API

Service-to-service authentication with Azure Data Lake Storage Gen1 using Python

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn about how to use the Python SDK to do service-to-service authentication with Azure Data Lake Storage Gen1. For end-user authentication with Data Lake Storage Gen1 using Python, see End-user authentication with Data Lake Storage Gen1 using Python.

Prerequisites

- **Python**. You can download Python from here. This article uses Python 3.6.2.
- **An Azure subscription**. See Get Azure free trial.
- **Create an Azure Active Directory "Web" Application**. You must have completed the steps in Service-to-service authentication with Data Lake Storage Gen1 using Azure Active Directory.

Install the modules

To work with Data Lake Storage Gen1 using Python, you need to install three modules.

- The `azure-mgmt-resource` module, which includes Azure modules for Active Directory, etc.
- The `azure-mgmt-datalake-store` module, which includes the Data Lake Storage Gen1 account management operations. For more information on this module, see Azure Data Lake Storage Gen1 Management module reference.
- The `azure-datalake-store` module, which includes the Data Lake Storage Gen1 filesystem operations. For more information on this module, see azure-datalake-store Filesystem module reference.

Use the following commands to install the modules.

```
pip install azure-mgmt-resource
pip install azure-mgmt-datalake-store
pip install azure-datalake-store
```

Create a new Python application

1. In the IDE of your choice create a new Python application, for example, **mysample.py**.
2. Add the following snippet to import the required modules

```
 ## Use this for Azure AD authentication
 from msrestazure.azure_active_directory import AADTokenCredentials

## Required for Data Lake Storage Gen1 account management
 from azure.mgmt.datalake.store import DataLakeStoreAccountManagementClient
 from azure.mgmt.datalake.store.models import DataLakeStoreAccount

 ## Required for Data Lake Storage Gen1 filesystem management
 from azure.datalake.store import core, lib, multithread

 # Common Azure imports
import adal
 from azure.mgmt.resource.resources import ResourceManagementClient
 from azure.mgmt.resource.resources.models import ResourceGroup

 ## Use these as needed for your application
 import logging, getpass, pprint, uuid, time
```

3. Save changes to mysample.py.

Service-to-service authentication with client secret for account management

Use this snippet to authenticate with Azure AD for account management operations on Data Lake Storage Gen1 such as create a Data Lake Storage Gen1 account, delete a Data Lake Storage Gen1 account, etc. The following snippet can be used to authenticate your application non-interactively, using the client secret for an application / service principal of an existing Azure AD "Web App" application.

```
authority_host_uri = 'https://login.microsoftonline.com'
tenant = '<TENANT>'
authority_uri = authority_host_uri + '/' + tenant
RESOURCE = 'https://management.core.windows.net/'
client_id = '<CLIENT_ID>'
client_secret = '<CLIENT_SECRET>'

context = adal.AuthenticationContext(authority_uri, api_version=None)
mgmt_token = context.acquire_token_with_client_credentials(RESOURCE, client_id, client_secret)
armCreds = AADTokenCredentials(mgmt_token, client_id, resource=RESOURCE)
```

Service-to-service authentication with client secret for filesystem operations

Use the following snippet to authenticate with Azure AD for filesystem operations on Data Lake Storage Gen1 such as create folder, upload file, etc. The following snippet can be used to authenticate your application non-interactively, using the client secret for an application / service principal. Use this with an existing Azure AD "Web App" application.

```
tenant = '<TENANT>'
RESOURCE = 'https://datalake.azure.net/'
client_id = '<CLIENT_ID>'
client_secret = '<CLIENT_SECRET>'

adlCreds = lib.auth(tenant_id = tenant,
                client_secret = client_secret,
                client_id = client_id,
                resource = RESOURCE)
```

Next steps

In this article, you learned how to use service-to-service authentication to authenticate with Data Lake Storage Gen1 using Python. You can now look at the following articles that talk about how to use Python to work with Data Lake Storage Gen1.

- Account management operations on Data Lake Storage Gen1 using Python
- Data operations on Data Lake Storage Gen1 using Python

Account management operations on Azure Data Lake Storage Gen1 using .NET SDK

10/3/2022 • 3 minutes to read • Edit Online

In this article, you learn how to perform account management operations on Azure Data Lake Storage Gen1 using .NET SDK. Account management operations include creating a Data Lake Storage Gen1 account, listing the accounts in an Azure subscription, deleting the accounts, etc.

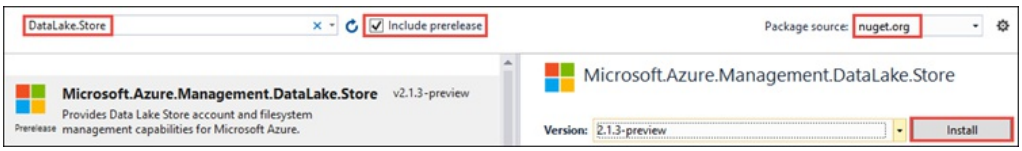
For instructions on how to perform data management operations on Data Lake Storage Gen1 using .NET SDK, see Filesystem operations on Data Lake Storage Gen1 using .NET SDK.

Prerequisites

- **Visual Studio 2013 or above**. The instructions below use Visual Studio 2019.
- **An Azure subscription**. See Get Azure free trial.

Create a .NET application

1. In Visual Studio, select the **File** menu, **New**, and then **Project**.
2. Choose **Console App (.NET Framework)**, and then select **Next**.
3. In **Project name**, enter `CreateADLApplication`, and then select **Create**.
4. Add the NuGet packages to your project.
   a. Right-click the project name in the Solution Explorer and click **Manage NuGet Packages**.
   b. In the **NuGet Package Manager** tab, make sure that **Package source** is set to **nuget.org** and that **Include prerelease** check box is selected.
   c. Search for and install the following NuGet packages:
      - `Microsoft.Azure.Management.DataLake.Store` - This tutorial uses v2.1.3-preview.
      - `Microsoft.Rest.ClientRuntime.Azure.Authentication` - This tutorial uses v2.2.12.

   d. Close the **NuGet Package Manager**.
5. Open **Program.cs**, delete the existing code, and then include the following statements to add references to namespaces.

```
using System;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Collections.Generic;
using System.Security.Cryptography.X509Certificates; // Required only if you are using an Azure AD
application created with certificates

using Microsoft.Rest;
using Microsoft.Rest.Azure.Authentication;
using Microsoft.Azure.Management.DataLake.Store;
using Microsoft.Azure.Management.DataLake.Store.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
```

6. Declare the variables and provide the values for placeholders. Also, make sure the local path and file name you provide exist on the computer.

```
namespace SdkSample
{
    class Program
    {
        private static DataLakeStoreAccountManagementClient _adlsClient;

        private static string _adlsAccountName;
        private static string _resourceGroupName;
        private static string _location;
        private static string _subId;

        private static void Main(string[] args)
        {
            _adlsAccountName = "<DATA-LAKE-STORAGE-GEN1-NAME>.azuredatalakestore.net";
            _resourceGroupName = "<RESOURCE-GROUP-NAME>";
            _location = "East US 2";
            _subId = "<SUBSCRIPTION-ID>";
        }
    }
}
```

In the remaining sections of the article, you can see how to use the available .NET methods to perform operations such as authentication, file upload, etc.

Authentication

- For end-user authentication for your application, see End-user authentication with Data Lake Storage Gen1 using .NET SDK.
- For service-to-service authentication for your application, see Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK.

Create client object

The following snippet creates the Data Lake Storage Gen1 account client object, which is used to issue account management requests to the service, such as create account, delete account, etc.

```
// Create client objects and set the subscription ID
_adlsClient = new DataLakeStoreAccountManagementClient(armCreds) { SubscriptionId = _subId };
```

Create a Data Lake Storage Gen1 account

The following snippet creates a Data Lake Storage Gen1 account in the Azure subscription you provided while creating the Data Lake Storage Gen1 account client object.

```
// Create Data Lake Storage Gen1 account
var adlsParameters = new DataLakeStoreAccount(location: _location);
_adlsClient.Account.Create(_resourceGroupName, _adlsAccountName, adlsParameters);
```

List all Data Lake Storage Gen1 accounts within a subscription

Add the following method to your class definition. The following snippet lists all Data Lake Storage Gen1 accounts within a given Azure subscription.

```
// List all Data Lake Storage Gen1 accounts within the subscription
public static List<DataLakeStoreAccountBasic> ListAdlStoreAccounts()
{
    var response = _adlsClient.Account.List(_adlsAccountName);
    var accounts = new List<DataLakeStoreAccountBasic>(response);

    while (response.NextPageLink != null)
    {
        response = _adlsClient.Account.ListNext(response.NextPageLink);
        accounts.AddRange(response);
    }

    return accounts;
}
```

Delete a Data Lake Storage Gen1 account

The following snippet deletes the Data Lake Storage Gen1 account you created earlier.

```
// Delete Data Lake Storage Gen1 account
_adlsClient.Account.Delete(_resourceGroupName, _adlsAccountName);
```

See also

- Filesystem operations on Data Lake Storage Gen1 using .NET SDK
- Data Lake Storage Gen1 .NET SDK Reference

Next steps

- Secure data in Data Lake Storage Gen1

Account management operations on Azure Data Lake Storage Gen1 using REST API

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn how to perform account management operations on Azure Data Lake Storage Gen1 using the REST API. Account management operations include creating a Data Lake Storage Gen1 account, deleting a Data Lake Storage Gen1 account, etc. For instructions on how to perform filesystem operations on Data Lake Storage Gen1 using REST API, see Filesystem operations on Data Lake Storage Gen1 using REST API.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **cURL**. This article uses cURL to demonstrate how to make REST API calls against a Data Lake Storage Gen1 account.

How do I authenticate using Azure Active Directory?

You can use two approaches to authenticate using Azure Active Directory.

- For end-user authentication for your application (interactive), see End-user authentication with Data Lake Storage Gen1 using .NET SDK.
- For service-to-service authentication for your application (non-interactive), see Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK.

Create a Data Lake Storage Gen1 account

This operation is based on the REST API call defined here.

Use the following cURL command. Replace **<yourstoragegen1name>** with your Data Lake Storage Gen1 name.

```
curl -i -X PUT -H "Authorization: Bearer <REDACTED>" -H "Content-Type: application/json"
https://management.azure.com/subscriptions/{subscription-id}/resourceGroups/{resource-group-
name}/providers/Microsoft.DataLakeStore/accounts/<yourstoragegen1name>?api-version=2015-10-01-preview -
d@"C:\temp\input.json"
```

In the above command, replace < `REDACTED` > with the authorization token you retrieved earlier. The request payload for this command is contained in the **input.json** file that is provided for the `-d` parameter above. The contents of the input.json file resemble the following snippet:

```
{
"location": "eastus2",
"tags": {
    "department": "finance"
    },
"properties": {}
}
```

Delete a Data Lake Storage Gen1 account

This operation is based on the REST API call defined here.

Use the following cURL command to delete a Data Lake Storage Gen1 account. Replace **<yourstoragegen1name>** with your Data Lake Storage Gen1 account name.

```
curl -i -X DELETE -H "Authorization: Bearer <REDACTED>"
https://management.azure.com/subscriptions/{subscription-id}/resourceGroups/{resource-group-
name}/providers/Microsoft.DataLakeStore/accounts/<yourstoragegen1name>?api-version=2015-10-01-preview
```

You should see an output like the following snippet:

```
HTTP/1.1 200 OK
...
...
```

Next steps

- Filesystem operations on Data Lake Storage Gen1 using REST API.

See also

- Azure Data Lake Storage Gen1 REST API Reference
- Open Source Big Data applications compatible with Azure Data Lake Storage Gen1

Account management operations on Azure Data Lake Storage Gen1 using Python

10/3/2022 • 2 minutes to read • Edit Online

Learn how to use the Python SDK for Azure Data Lake Storage Gen1 to perform basic account management operations such as create a Data Lake Storage Gen1 account, list the Data Lake Storage Gen1 accounts, etc. For instructions on how to perform filesystem operations on Data Lake Storage Gen1 using Python, see Filesystem operations on Data Lake Storage Gen1 using Python.

Prerequisites

- **Python**. You can download Python from here. This article uses Python 3.6.2.
- **An Azure subscription**. See Get Azure free trial.
- **An Azure resource group**. For instructions, see Create an Azure resource group.

Install the modules

To work with Data Lake Storage Gen1 using Python, you need to install three modules.

- The `azure-mgmt-resource` module, which includes Azure modules for Active Directory, etc.
- The `azure-mgmt-datalake-store` module, which includes the Azure Data Lake Storage Gen1 account management operations. For more information on this module, see Azure Data Lake Storage Gen1 Management module reference.
- The `azure-datalake-store` module, which includes the Azure Data Lake Storage Gen1 filesystem operations. For more information on this module, see azure-datalake-store filesystem module reference.

Use the following commands to install the modules.

```
pip install azure-identity
pip install azure-mgmt-resource
pip install azure-mgmt-datalake-store
pip install azure-datalake-store
```

Create a new Python application

1. In the IDE of your choice create a new Python application, for example, **mysample.py**.
2. Add the following snippet to import the required modules

```
# Acquire a credential object for the app identity. When running in the cloud,
# DefaultAzureCredential uses the app's managed identity (MSI) or user-assigned service principal.
# When run locally, DefaultAzureCredential relies on environment variables named
# AZURE_CLIENT_ID, AZURE_CLIENT_SECRET, and AZURE_TENANT_ID.
from azure.identity import DefaultAzureCredential

## Required for Data Lake Storage Gen1 account management
from azure.mgmt.datalake.store import DataLakeStoreAccountManagementClient
from azure.mgmt.datalake.store.models import CreateDataLakeStoreAccountParameters

## Required for Data Lake Storage Gen1 filesystem management
from azure.datalake.store import core, lib, multithread

# Common Azure imports
import adal
from azure.mgmt.resource.resources import ResourceManagementClient
from azure.mgmt.resource.resources.models import ResourceGroup

# Use these as needed for your application
import logging, getpass, pprint, uuid, time
```

3. Save changes to mysample.py.

Authentication

In this section, we talk about the different ways to authenticate with Azure AD. The options available are:

- For end-user authentication for your application, see End-user authentication with Data Lake Storage Gen1 using Python.
- For service-to-service authentication for your application, see Service-to-service authentication with Data Lake Storage Gen1 using Python.

Create client and Data Lake Storage Gen1 account

The following snippet first creates the Data Lake Storage Gen1 account client. It uses the client object to create a Data Lake Storage Gen1 account. Finally, the snippet creates a filesystem client object.

```
## Declare variables
subscriptionId = 'FILL-IN-HERE'
adlsAccountName = 'FILL-IN-HERE'
resourceGroup = 'FILL-IN-HERE'
location = 'eastus2'
credential = DefaultAzureCredential()

## Create Data Lake Storage Gen1 account management client object
adlsAcctClient = DataLakeStoreAccountManagementClient(credential, subscription_id=subscriptionId)

## Create a Data Lake Storage Gen1 account
adlsAcctResult = adlsAcctClient.accounts.begin_create(
 resourceGroup,
 adlsAccountName,
 CreateDataLakeStoreAccountParameters(
  location=location
 )
)
```

List the Data Lake Storage Gen1 accounts

```
## List the existing Data Lake Storage Gen1 accounts
result_list_response = adlsAcctClient.accounts.list()
result_list = list(result_list_response)
for items in result_list:
    print(items)
```

Delete the Data Lake Storage Gen1 account

```
## Delete an existing Data Lake Storage Gen1 account
adlsAcctClient.accounts.begin_delete(resourceGroup, adlsAccountName)
```

Next steps

- Filesystem operations on Data Lake Storage Gen1 using Python.

See also

- azure-datalake-store Python (Filesystem) reference
- Open Source Big Data applications compatible with Azure Data Lake Storage Gen1

Filesystem operations on Data Lake Storage Gen1 using the .NET SDK

10/3/2022 • 4 minutes to read • Edit Online

In this article, you learn how to perform filesystem operations on Data Lake Storage Gen1 using the .NET SDK. Filesystem operations include creating folders in a Data Lake Storage Gen1 account, uploading files, downloading files, etc.

For instructions on how to do account management operations on Data Lake Storage Gen1 using the .NET SDK, see Account management operations on Data Lake Storage Gen1 using .NET SDK.

Prerequisites

- **Visual Studio 2013 or above**. The instructions in this article use Visual Studio 2019.
- **An Azure subscription**. See Get Azure free trial.
- **Azure Data Lake Storage Gen1 account**. For instructions on how to create an account, see Get started with Azure Data Lake Storage Gen1.

Create a .NET application

The code sample available on GitHub walks you through the process of creating files in the store, concatenating files, downloading a file, and deleting some files in the store. This section of the article walks you through the main parts of the code.

1. In Visual Studio, select the **File** menu, **New**, and then **Project**.
2. Choose **Console App (.NET Framework)**, and then select **Next**.
3. In **Project name**, enter `CreateADLApplication`, and then select **Create**.
4. Add the NuGet packages to your project.
   a. Right-click the project name in the Solution Explorer and click **Manage NuGet Packages**.
   b. In the **NuGet Package Manager** tab, make sure that **Package source** is set to **nuget.org**. Also, make sure the **Include prerelease** check box is selected.
   c. Search for and install the following NuGet packages:
      - `Microsoft.Azure.DataLake.Store` - This article uses v1.0.0.
      - `Microsoft.Rest.ClientRuntime.Azure.Authentication` - This article uses v2.3.1.

      Close the **NuGet Package Manager**.
5. Open **Program.cs**, delete the existing code, and then include the following statements to add references to namespaces.

```
using System;
using System.IO;using System.Threading;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using System.Security.Cryptography.X509Certificates; // Required only if you're using an Azure AD
application created with certificates

using Microsoft.Rest;
using Microsoft.Rest.Azure.Authentication;
using Microsoft.Azure.DataLake.Store;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
```

6. Declare the variables as shown below, and provide the values for the placeholders. Also, make sure the local path and file name you provide here exist on the computer.

```
namespace SdkSample
{
    class Program
    {
        private static string _adlsg1AccountName = "<DATA-LAKE-STORAGE-GEN1-
NAME>.azuredatalakestore.net";
    }
}
```

In the remaining sections of the article, you can see how to use the available .NET methods to do operations such as authentication, file upload, etc.

Authentication

- For end-user authentication for your application, see End-user authentication with Data Lake Storage Gen1 using .NET SDK.
- For service-to-service authentication for your application, see Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK.

Create client object

The following snippet creates the Data Lake Storage Gen1 filesystem client object, which is used to issue requests to the service.

```
// Create client objects
AdlsClient client = AdlsClient.CreateClient(_adlsg1AccountName, adlCreds);
```

Create a file and directory

Add the following snippet to your application. This snippet adds a file and any parent directory that does not exist.

```
// Create a file - automatically creates any parent directories that don't exist
// The AdlsOutputStream preserves record boundaries - it does not break records while writing to the store

using (var stream = client.CreateFile(fileName, IfExists.Overwrite))
{
    byte[] textByteArray = Encoding.UTF8.GetBytes("This is test data to write.\r\n");
    stream.Write(textByteArray, 0, textByteArray.Length);

    textByteArray = Encoding.UTF8.GetBytes("This is the second line.\r\n");
    stream.Write(textByteArray, 0, textByteArray.Length);
}
```

Append to a file

The following snippet appends data to an existing file in Data Lake Storage Gen1 account.

```
// Append to existing file

using (var stream = client.GetAppendStream(fileName))
{
    byte[] textByteArray = Encoding.UTF8.GetBytes("This is the added line.\r\n");
    stream.Write(textByteArray, 0, textByteArray.Length);
}
```

Read a file

The following snippet reads the contents of a file in Data Lake Storage Gen1.

```
//Read file contents

using (var readStream = new StreamReader(client.GetReadStream(fileName)))
{
    string line;
    while ((line = readStream.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

Get file properties

The following snippet returns the properties associated with a file or a directory.

```
// Get file properties
var directoryEntry = client.GetDirectoryEntry(fileName);
PrintDirectoryEntry(directoryEntry);
```

The definition of the `PrintDirectoryEntry` method is available as part of the sample on GitHub.

Rename a file

The following snippet renames an existing file in a Data Lake Storage Gen1 account.

```
// Rename a file
string destFilePath = "/Test/testRenameDest3.txt";
client.Rename(fileName, destFilePath, true);
```

Enumerate a directory

The following snippet enumerates directories in a Data Lake Storage Gen1 account.

```
// Enumerate directory
foreach (var entry in client.EnumerateDirectory("/Test"))
{
    PrintDirectoryEntry(entry);
}
```

The definition of the `PrintDirectoryEntry` method is available as part of the sample on GitHub.

Delete directories recursively

The following snippet deletes a directory, and all its subdirectories, recursively.

```
// Delete a directory and all its subdirectories and files
client.DeleteRecursive("/Test");
```

Samples

Here are a few samples that show how to use the Data Lake Storage Gen1 Filesystem SDK.

- Basic sample on GitHub
- Advanced sample on GitHub

See also

- Account management operations on Data Lake Storage Gen1 using .NET SDK
- Data Lake Storage Gen1 .NET SDK Reference

Next steps

- Secure data in Data Lake Storage Gen1

Filesystem operations on Azure Data Lake Storage Gen1 using Java SDK

10/3/2022 • 5 minutes to read • Edit Online

Learn how to use the Azure Data Lake Storage Gen1 Java SDK to perform basic operations such as create folders, upload and download data files, etc. For more information about Data Lake Storage Gen1, see Azure Data Lake Storage Gen1.

You can access the Java SDK API docs for Data Lake Storage Gen1 at Azure Data Lake Storage Gen1 Java API docs.

Prerequisites

- Java Development Kit (JDK 7 or higher, using Java version 1.7 or higher)
- Data Lake Storage Gen1 account. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal.
- Maven. This tutorial uses Maven for build and project dependencies. Although it is possible to build without using a build system like Maven or Gradle, these systems make is much easier to manage dependencies.
- (Optional) And IDE like IntelliJ IDEA or Eclipse or similar.

Create a Java application

The code sample available on GitHub walks you through the process of creating files in the store, concatenating files, downloading a file, and deleting some files in the store. This section of the article walks you through the main parts of the code.

1. Create a Maven project using mvn archetype from the command line or using an IDE. For instructions on how to create a Java project using IntelliJ, see here. For instructions on how to create a project using Eclipse, see here.

2. Add the following dependencies to your Maven **pom.xml** file. Add the following snippet before the **</project>** tag:

```
<dependencies>
    <dependency>
    <groupId>com.microsoft.azure</groupId>
    <artifactId>azure-data-lake-store-sdk</artifactId>
    <version>2.1.5</version>
    </dependency>
    <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-nop</artifactId>
    <version>1.7.21</version>
    </dependency>
</dependencies>
```

   The first dependency is to use the Data Lake Storage Gen1 SDK (`azure-data-lake-store-sdk`) from the maven repository. The second dependency is to specify the logging framework (`slf4j-nop`) to use for this application. The Data Lake Storage Gen1 SDK uses SLF4J logging façade, which lets you choose from a number of popular logging frameworks, like Log4j, Java logging, Logback, etc., or no logging. For this example, we disable logging, hence we use the **slf4j-nop** binding. To use other logging options in your

app, see here.

3. Add the following import statements to your application.

```
import com.microsoft.azure.datalake.store.ADLException;
import com.microsoft.azure.datalake.store.ADLStoreClient;
import com.microsoft.azure.datalake.store.DirectoryEntry;
import com.microsoft.azure.datalake.store.IfExists;
import com.microsoft.azure.datalake.store.oauth2.AccessTokenProvider;
import com.microsoft.azure.datalake.store.oauth2.ClientCredsTokenProvider;

import java.io.*;
import java.util.Arrays;
import java.util.List;
```

Authentication

- For end-user authentication for your application, see End-user-authentication with Data Lake Storage Gen1 using Java.
- For service-to-service authentication for your application, see Service-to-service authentication with Data Lake Storage Gen1 using Java.

Create a Data Lake Storage Gen1 client

Creating an ADLStoreClient object requires you to specify the Data Lake Storage Gen1 account name and the token provider you generated when you authenticated with Data Lake Storage Gen1 (see Authentication section). The Data Lake Storage Gen1 account name needs to be a fully qualified domain name. For example, replace **FILL-IN-HERE** with something like **mydatalakestoragegen1.azuredatalakestore.net**.

```
private static String accountFQDN = "FILL-IN-HERE";  // full account FQDN, not just the account name
ADLStoreClient client = ADLStoreClient.createClient(accountFQDN, provider);
```

The code snippets in the following sections contain examples of some common filesystem operations. You can look at the full Data Lake Storage Gen1 Java SDK API docs of the **ADLStoreClient** object to see other operations.

Create a directory

The following snippet creates a directory structure in the root of the Data Lake Storage Gen1 account you specified.

```
// create directory
client.createDirectory("/a/b/w");
System.out.println("Directory created.");
```

Create a file

The following snippet creates a file (c.txt) in the directory structure and writes some data to the file.

```
// create file and write some content
String filename = "/a/b/c.txt";
OutputStream stream = client.createFile(filename, IfExists.OVERWRITE  );
PrintStream out = new PrintStream(stream);
for (int i = 1; i <= 10; i++) {
    out.println("This is line #" + i);
    out.format("This is the same line (%d), but using formatted output. %n", i);
}
out.close();
System.out.println("File created.");
```

You can also create a file (d.txt) using byte arrays.

```
// create file using byte arrays
stream = client.createFile("/a/b/d.txt", IfExists.OVERWRITE);
byte[] buf = getSampleContent();
stream.write(buf);
stream.close();
System.out.println("File created using byte array.");
```

The definition for `getSampleContent` function used in the preceding snippet is available as part of the sample on GitHub.

Append to a file

The following snippet appends content to an existing file.

```
// append to file
stream = client.getAppendStream(filename);
stream.write(getSampleContent());
stream.close();
System.out.println("File appended.");
```

The definition for `getSampleContent` function used in the preceding snippet is available as part of the sample on GitHub.

Read a file

The following snippet reads content from a file in a Data Lake Storage Gen1 account.

```
// Read File
InputStream in = client.getReadStream(filename);
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
String line;
while ( (line = reader.readLine()) != null) {
    System.out.println(line);
}
reader.close();
System.out.println();
System.out.println("File contents read.");
```

Concatenate files

The following snippet concatenates two files in a Data Lake Storage Gen1 account. If successful, the concatenated file replaces the two existing files.

```
// concatenate the two files into one
List<String> fileList = Arrays.asList("/a/b/c.txt", "/a/b/d.txt");
client.concatenateFiles("/a/b/f.txt", fileList);
System.out.println("Two files concatenated into a new file.");
```

Rename a file

The following snippet renames a file in a Data Lake Storage Gen1 account.

```
//rename the file
client.rename("/a/b/f.txt", "/a/b/g.txt");
System.out.println("New file renamed.");
```

Get metadata for a file

The following snippet retrieves the metadata for a file in a Data Lake Storage Gen1 account.

```
// get file metadata
DirectoryEntry ent = client.getDirectoryEntry(filename);
printDirectoryInfo(ent);
System.out.println("File metadata retrieved.");
```

Set permissions on a file

The following snippet sets permissions on the file that you created in the previous section.

```
// set file permission
client.setPermission(filename, "744");
System.out.println("File permission set.");
```

List directory contents

The following snippet lists the contents of a directory, recursively.

```
// list directory contents
List<DirectoryEntry> list = client.enumerateDirectory("/a/b", 2000);
System.out.println("Directory listing for directory /a/b:");
for (DirectoryEntry entry : list) {
    printDirectoryInfo(entry);
}
System.out.println("Directory contents listed.");
```

The definition for `printDirectoryInfo` function used in the preceding snippet is available as part of the sample on GitHub.

Delete files and folders

The following snippet deletes the specified files and folders in a Data Lake Storage Gen1 account, recursively.

```
// delete directory along with all the subdirectories and files in it
client.deleteRecursive("/a");
System.out.println("All files and folders deleted recursively");
promptEnterKey();
```

Build and run the application

1. To run from within an IDE, locate and press the **Run** button. To run from Maven, use exec:exec.
2. To produce a standalone jar that you can run from command-line build the jar with all dependencies included, using the Maven assembly plugin. The pom.xml in the example source code on GitHub has an example.

Next steps

- Explore JavaDoc for the Java SDK
- Secure data in Data Lake Storage Gen1

Filesystem operations on Azure Data Lake Storage Gen1 using REST API

10/3/2022 • 3 minutes to read • Edit Online

In this article, you learn how to use WebHDFS REST APIs and Data Lake Storage Gen1 REST APIs to perform filesystem operations on Azure Data Lake Storage Gen1. For instructions on how to perform account management operations on Data Lake Storage Gen1 using REST API, see Account management operations on Data Lake Storage Gen1 using REST API.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **Azure Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal.
- **cURL**. This article uses cURL to demonstrate how to make REST API calls against a Data Lake Storage Gen1 account.

How do I authenticate using Azure Active Directory?

You can use two approaches to authenticate using Azure Active Directory.

- For end-user authentication for your application (interactive), see End-user authentication with Data Lake Storage Gen1 using .NET SDK.
- For service-to-service authentication for your application (non-interactive), see Service-to-service authentication with Data Lake Storage Gen1 using .NET SDK.

Create folders

This operation is based on the WebHDFS REST API call defined here.

Use the following cURL command. Replace **<yourstorename>** with your Data Lake Storage Gen1 account name.

```
curl -i -X PUT -H "Authorization: Bearer <REDACTED>" -d ""
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/?op=MKDIRS'
```

In the preceding command, replace < `REDACTED` > with the authorization token you retrieved earlier. This command creates a directory called **mytempdir** under the root folder of your Data Lake Storage Gen1 account.

If the operation completes successfully, you should see a response like the following snippet:

```
{"boolean":true}
```

List folders

This operation is based on the WebHDFS REST API call defined here.

Use the following cURL command. Replace **<yourstorename>** with your Data Lake Storage Gen1 account

name.

```
curl -i -X GET -H "Authorization: Bearer <REDACTED>"
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/?op=LISTSTATUS'
```

In the preceding command, replace < `REDACTED` > with the authorization token you retrieved earlier.

If the operation completes successfully, you should see a response like the following snippet:

```
{
"FileStatuses": {
    "FileStatus": [{
        "length": 0,
        "pathSuffix": "mytempdir",
        "type": "DIRECTORY",
        "blockSize": 268435456,
        "accessTime": 1458324719512,
        "modificationTime": 1458324719512,
        "replication": 0,
        "permission": "777",
        "owner": "<GUID>",
        "group": "<GUID>"
    }]
}
}
```

Upload data

This operation is based on the WebHDFS REST API call defined here.

Use the following cURL command. Replace **<yourstorename>** with your Data Lake Storage Gen1 account name.

```
curl -i -X PUT -L -T 'C:\temp\list.txt' -H "Authorization: Bearer <REDACTED>"
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/list.txt?op=CREATE'
```

In the preceding syntax **-T** parameter is the location of the file you are uploading.

The output is similar to the following snippet:

```
HTTP/1.1 307 Temporary Redirect
...
Location: https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/list.txt?op=CREATE&write=true
...
Content-Length: 0

HTTP/1.1 100 Continue

HTTP/1.1 201 Created
...
```

Read data

This operation is based on the WebHDFS REST API call defined here.

Reading data from a Data Lake Storage Gen1 account is a two-step process.

- You first submit a GET request against the endpoint

`https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/myinputfile.txt?op=OPEN`. This call returns a location to submit the next GET request to.

- You then submit the GET request against the endpoint `https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/myinputfile.txt?op=OPEN&read=true`. This call displays the contents of the file.

However, because there is no difference in the input parameters between the first and the second step, you can use the `-L` parameter to submit the first request. `-L` option essentially combines two requests into one and makes cURL redo the request on the new location. Finally, the output from all the request calls is displayed, like shown in the following snippet. Replace **<yourstorename>** with your Data Lake Storage Gen1 account name.

```
curl -i -L GET -H "Authorization: Bearer <REDACTED>"
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/myinputfile.txt?op=OPEN'
```

You should see an output similar to the following snippet:

```
HTTP/1.1 307 Temporary Redirect
...
Location: https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/somerandomfile.txt?
op=OPEN&read=true
...

HTTP/1.1 200 OK
...

Hello, Data Lake Store user!
```

Rename a file

This operation is based on the WebHDFS REST API call defined here.

Use the following cURL command to rename a file. Replace **<yourstorename>** with your Data Lake Storage Gen1 account name.

```
curl -i -X PUT -H "Authorization: Bearer <REDACTED>" -d ""
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/myinputfile.txt?
op=RENAME&destination=/mytempdir/myinputfile1.txt'
```

You should see an output similar to the following snippet:

```
HTTP/1.1 200 OK
...

{"boolean":true}
```

Delete a file

This operation is based on the WebHDFS REST API call defined here.

Use the following cURL command to delete a file. Replace **<yourstorename>** with your Data Lake Storage Gen1 account name.

```
curl -i -X DELETE -H "Authorization: Bearer <REDACTED>"
'https://<yourstorename>.azuredatalakestore.net/webhdfs/v1/mytempdir/myinputfile1.txt?op=DELETE'
```

You should see an output like the following:

```
HTTP/1.1 200 OK
...

{"boolean":true}
```

Next steps

- Account management operations on Data Lake Storage Gen1 using REST API.

See also

- Azure Data Lake Storage Gen1 REST API Reference
- Open Source Big Data applications compatible with Azure Data Lake Storage Gen1

Filesystem operations on Azure Data Lake Storage Gen1 using Python

10/3/2022 • 2 minutes to read • Edit Online

In this article, you learn how to use Python SDK to perform filesystem operations on Azure Data Lake Storage Gen1. For instructions on how to perform account management operations on Data Lake Storage Gen1 using Python, see Account management operations on Data Lake Storage Gen1 using Python.

Prerequisites

- **Python**. You can download Python from here. This article uses Python 3.6.2.
- **An Azure subscription**. See Get Azure free trial.
- **Azure Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal.

Install the modules

To work with Data Lake Storage Gen1 using Python, you need to install three modules.

- The `azure-mgmt-resource` module, which includes Azure modules for Active Directory, etc.
- The `azure-mgmt-datalake-store` module, which includes the Azure Data Lake Storage Gen1 account management operations. For more information on this module, see the azure-mgmt-datalake-store module reference.
- The `azure-datalake-store` module, which includes the Azure Data Lake Storage Gen1 filesystem operations. For more information on this module, see the azure-datalake-store file-system module reference.

Use the following commands to install the modules.

```
pip install azure-mgmt-resource
pip install azure-mgmt-datalake-store
pip install azure-datalake-store
```

Create a new Python application

1. In the IDE of your choice create a new Python application, for example, **mysample.py**.
2. Add the following lines to import the required modules

```
## Use this only for Azure AD service-to-service authentication
from azure.common.credentials import ServicePrincipalCredentials

## Use this only for Azure AD end-user authentication
from azure.common.credentials import UserPassCredentials

## Use this only for Azure AD multi-factor authentication
from msrestazure.azure_active_directory import AADTokenCredentials

## Required for Azure Data Lake Storage Gen1 account management
from azure.mgmt.datalake.store import DataLakeStoreAccountManagementClient
from azure.mgmt.datalake.store.models import DataLakeStoreAccount

## Required for Azure Data Lake Storage Gen1 filesystem management
from azure.datalake.store import core, lib, multithread

## Common Azure imports
from azure.mgmt.resource.resources import ResourceManagementClient
from azure.mgmt.resource.resources.models import ResourceGroup

## Use these as needed for your application
import logging, getpass, pprint, uuid, time
```

3. Save changes to mysample.py.

Authentication

In this section, we talk about the different ways to authenticate with Azure AD. The options available are:

- For end-user authentication for your application, see End-user authentication with Data Lake Storage Gen1 using Python.
- For service-to-service authentication for your application, see Service-to-service authentication with Data Lake Storage Gen1 using Python.

Create filesystem client

The following snippet first creates the Data Lake Storage Gen1 account client. It uses the client object to create a Data Lake Storage Gen1 account. Finally, the snippet creates a filesystem client object.

```
## Declare variables
subscriptionId = 'FILL-IN-HERE'
adlsAccountName = 'FILL-IN-HERE'

## Create a filesystem client object
adlsFileSystemClient = core.AzureDLFileSystem(adlCreds, store_name=adlsAccountName)
```

Create a directory

```
## Create a directory
adlsFileSystemClient.mkdir('/mysampledirectory')
```

Upload a file

```
## Upload a file
multithread.ADLUploader(adlsFileSystemClient, lpath='C:\\data\\mysamplefile.txt',
rpath='/mysampledirectory/mysamplefile.txt', nthreads=64, overwrite=True, buffersize=4194304,
blocksize=4194304)
```

Download a file

```
## Download a file
multithread.ADLDownloader(adlsFileSystemClient, lpath='C:\\data\\mysamplefile.txt.out',
rpath='/mysampledirectory/mysamplefile.txt', nthreads=64, overwrite=True, buffersize=4194304,
blocksize=4194304)
```

Delete a directory

```
## Delete a directory
adlsFileSystemClient.rm('/mysampledirectory', recursive=True)
```

Next steps

- Account management operations on Data Lake Storage Gen1 using Python.

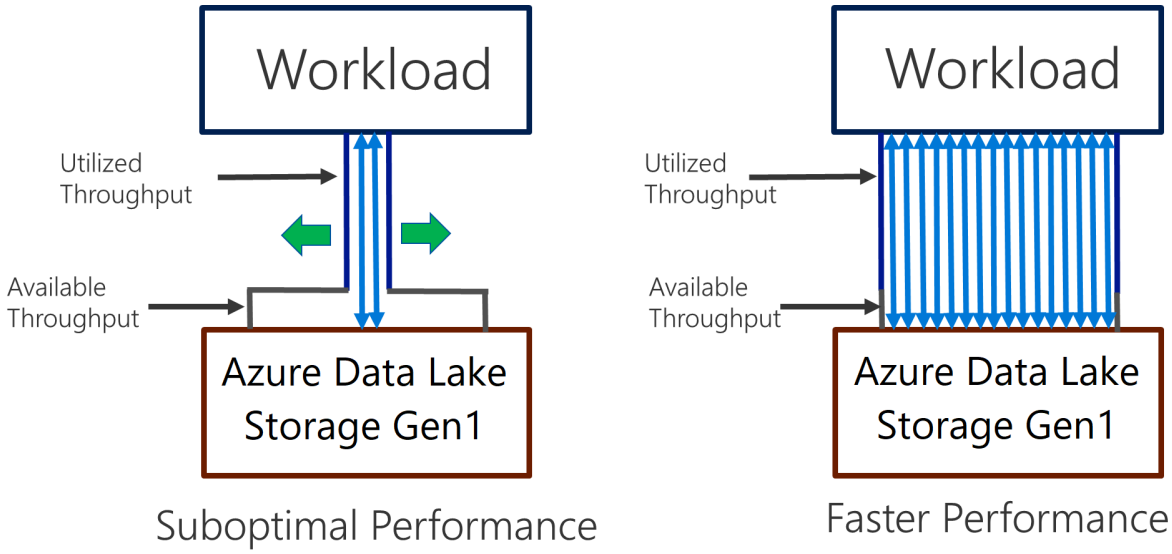
See also

- Azure Data Lake Storage Gen1 Python (Filesystem) Reference
- Open Source Big Data applications compatible with Azure Data Lake Storage Gen1

Tune Azure Data Lake Storage Gen1 for performance

10/3/2022 • 6 minutes to read • Edit Online

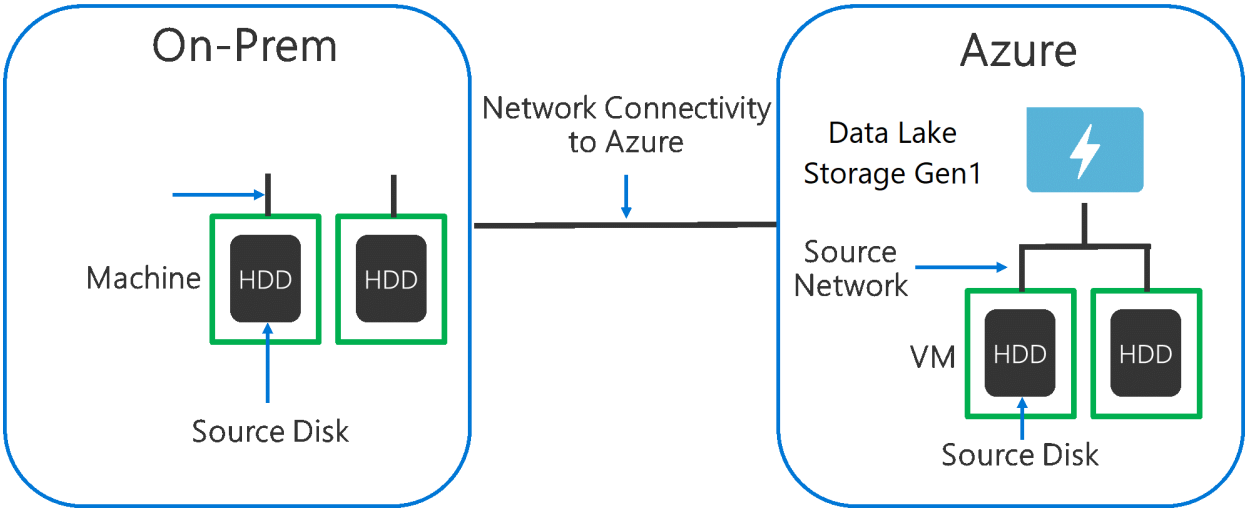
Data Lake Storage Gen1 supports high-throughput for I/O intensive analytics and data movement. In Data Lake Storage Gen1, using all available throughput – the amount of data that can be read or written per second – is important to get the best performance. This is achieved by performing as many reads and writes in parallel as possible.



Data Lake Storage Gen1 can scale to provide the necessary throughput for all analytics scenario. By default, a Data Lake Storage Gen1 account provides automatically enough throughput to meet the needs of a broad category of use cases. For the cases where customers run into the default limit, the Data Lake Storage Gen1 account can be configured to provide more throughput by contacting Microsoft support.

Data ingestion

When ingesting data from a source system to Data Lake Storage Gen1, it's important to consider that the source hardware, source network hardware, and network connectivity to Data Lake Storage Gen1 can be the bottleneck.



It's important to ensure that the data movement is not affected by these factors.

Source hardware

Whether you're using on-premises machines or VMs in Azure, you should carefully select the appropriate hardware. For Source Disk Hardware, prefer SSDs to HDDs and pick disk hardware with faster spindles. For Source Network Hardware, use the fastest NICs possible. On Azure, we recommend Azure D14 VMs that have the appropriately powerful disk and networking hardware.

Network connectivity to Data Lake Storage Gen1

The network connectivity between your source data and Data Lake Storage Gen1 can sometimes be the bottleneck. When your source data is On-Premises, consider using a dedicated link with Azure ExpressRoute . If your source data is in Azure, the performance will be best when the data is in the same Azure region as the Data Lake Storage Gen1 account.

Configure data ingestion tools for maximum parallelization

After you've addressed the source hardware and network connectivity bottlenecks, you're ready to configure your ingestion tools. The following table summarizes the key settings for several popular ingestion tools and provides in-depth performance tuning articles for them. To learn more about which tool to use for your scenario, visit this article.

TOOL	SETTINGS	MORE DETAILS
PowerShell	PerFileThreadCount, ConcurrentFileCount	Link
AdlCopy	Azure Data Lake Analytics units	Link
DistCp	-m (mapper)	Link
Azure Data Factory	parallelCopies	Link
Sqoop	fs.azure.block.size, -m (mapper)	Link

Structure your data set

When data is stored in Data Lake Storage Gen1, the file size, number of files, and folder structure affect performance. The following section describes best practices in these areas.

File size

Typically, analytics engines such as HDInsight and Azure Data Lake Analytics have a per-file overhead. 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. As a rule of thumb, organize data sets in files of 256 MB or larger. In some cases such as images and binary data, it is not possible to process them in parallel. In these cases, it is recommended to keep individual files under 2 GB.

Sometimes, data pipelines have limited control over the raw data that has lots of small files. It is recommended to have a "cooking" process that generates larger files to use for downstream applications.

Organize time-series data in folders

For Hive and ADLA 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. The following 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.

Optimize I/O intensive jobs on Hadoop and Spark workloads on HDInsight

Jobs fall into one of the following three categories:

- **CPU intensive.** These jobs have long computation times with minimal I/O times. Examples include machine learning and natural language processing jobs.
- **Memory intensive.** These jobs use lots of memory. Examples include PageRank and real-time analytics jobs.
- **I/O intensive.** These jobs spend most of their time doing I/O. A common example is a copy job that does only read and write operations. Other examples include data preparation jobs that read numerous data, performs some data transformation, and then writes the data back to the store.

The following guidance is only applicable to I/O intensive jobs.

General considerations for an HDInsight cluster

- **HDInsight versions.** For best performance, use the latest release of HDInsight.
- **Regions.** Place the Data Lake Storage Gen1 account in the same region as the HDInsight cluster.

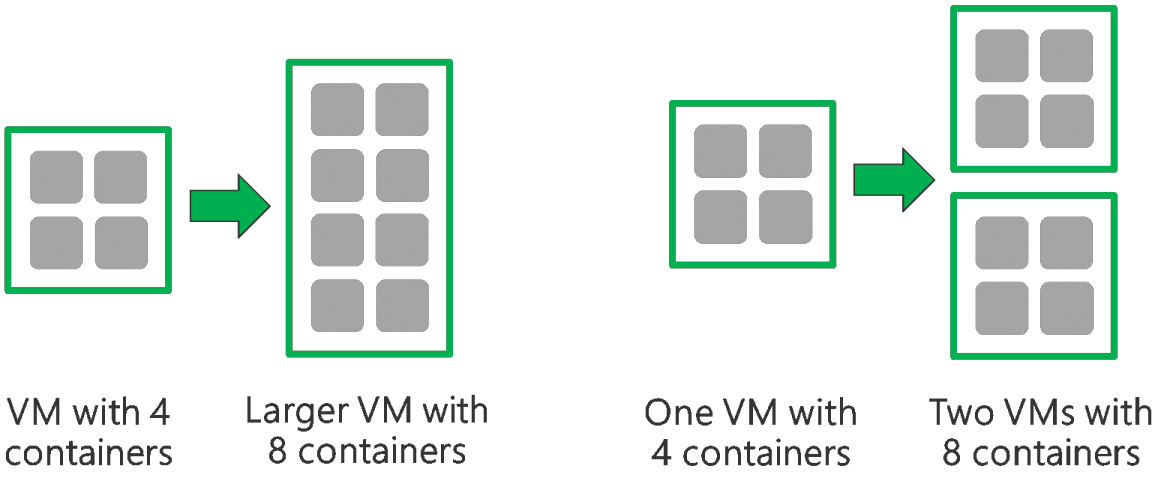
An HDInsight cluster is composed of two head nodes and some worker nodes. Each worker node provides a specific number of cores and memory, which is determined by the VM-type. When running a job, YARN is the resource negotiator that allocates the available memory and cores to create containers. Each container runs the tasks needed to complete the job. Containers run in parallel to process tasks quickly. Therefore, performance is improved by running as many parallel containers as possible.

There are three layers within an HDInsight cluster that can be tuned to increase the number of containers and use all available throughput.

- **Physical layer**
- **YARN layer**
- **Workload layer**

Physical layer

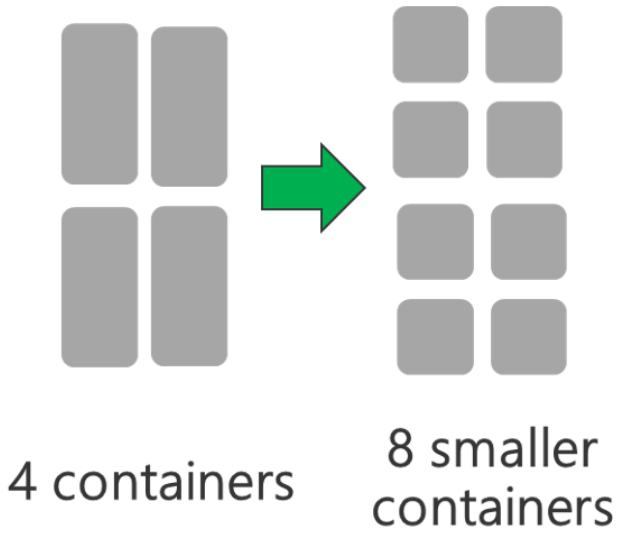
Run cluster with more nodes and/or larger sized VMs. A larger cluster will enable you to run more YARN containers as shown in the picture below.



Use VMs with more network bandwidth. The amount of network bandwidth can be a bottleneck if there is less network bandwidth than Data Lake Storage Gen1 throughput. Different VMs will have varying network bandwidth sizes. Choose a VM-type that has the largest possible network bandwidth.

YARN layer

Use smaller YARN containers. Reduce the size of each YARN container to create more containers with the same amount of resources.

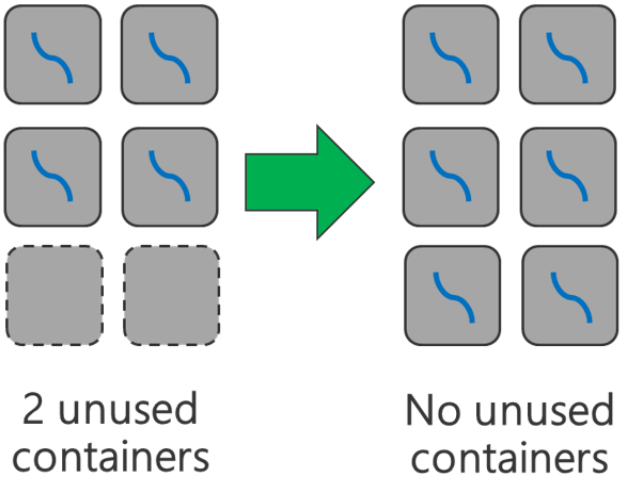


Depending on your workload, there will always be a minimum YARN container size that is needed. If you pick too small a container, your jobs will run into out-of-memory issues. Typically YARN containers should be no smaller than 1 GB. It's common to see 3 GB YARN containers. For some workloads, you may need larger YARN containers.

Increase cores per YARN container. Increase the number of cores allocated to each container to increase the number of parallel tasks that run in each container. This works for applications like Spark, which run multiple tasks per container. For applications like Hive that run a single thread in each container, it's better to have more containers rather than more cores per container.

Workload layer

Use all available containers. Set the number of tasks to be equal or larger than the number of available containers so that all resources are used.



Failed tasks are costly. If each task has a large amount of data to process, then failure of a task results in an expensive retry. Therefore, it's better to create more tasks, each of which processes a small amount of data.

In addition to the general guidelines above, each application has different parameters available to tune for that specific application. The table below lists some of the parameters and links to get started with performance tuning for each application.

WORKLOAD	PARAMETER TO SET TASKS
Spark on HDInsight	• Num-executors • Executor-memory • Executor-cores
Hive on HDInsight	• hive.tez.container.size
MapReduce on HDInsight	• Mapreduce.map.memory • Mapreduce.job.maps • Mapreduce.reduce.memory • Mapreduce.job.reduces
Storm on HDInsight	• Number of worker processes • Number of spout executor instances • Number of bolt executor instances • Number of spout tasks • Number of bolt tasks

See also

- Overview of Azure Data Lake Storage Gen1
- Get Started with Azure Data Lake Analytics

Performance tuning guidance for using PowerShell with Azure Data Lake Storage Gen1

10/3/2022 • 4 minutes to read • Edit Online

This article describes the properties that you can tune to get better performance while using PowerShell to work with Data Lake Storage Gen1.

> **NOTE**
>
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Performance-related properties

PROPERTY	DEFAULT	DESCRIPTION
PerFileThreadCount	10	This parameter enables you to choose the number of parallel threads for uploading or downloading each file. This number represents the max threads that can be allocated per file, but you may get fewer threads depending on your scenario (for example, if you are uploading a 1-KB file, you get one thread even if you ask for 20 threads).
ConcurrentFileCount	10	This parameter is specifically for uploading or downloading folders. This parameter determines the number of concurrent files that can be uploaded or downloaded. This number represents the maximum number of concurrent files that can be uploaded or downloaded at one time, but you may get less concurrency depending on your scenario (for example, if you are uploading two files, you get two concurrent files uploads even if you ask for 15).

Example:

This command downloads files from Data Lake Storage Gen1 to the user's local drive using 20 threads per file and 100 concurrent files.

```
Export-AzDataLakeStoreItem -AccountName "Data Lake Storage Gen1 account name" `
    -PerFileThreadCount 20 `
 -ConcurrentFileCount 100 `
 -Path /Powershell/100GB `
 -Destination C:\Performance\ `
 -Force `
 -Recurse
```

How to determine property values

The next question you might have is how to determine what value to provide for the performance-related properties. Here's some guidance that you can use.

- **Step 1: Determine the total thread count** - Start by calculating the total thread count to use. As a general guideline, you should use six threads for each physical core.

  ```
  Total thread count = total physical cores * 6
  ```

  Example:

  Assuming you are running the PowerShell commands from a D14 VM that has 16 cores

  ```
  Total thread count = 16 cores * 6 = 96 threads
  ```

- **Step 2: Calculate PerFileThreadCount** - We calculate our PerFileThreadCount based on the size of the files. For files smaller than 2.5 GB, there is no need to change this parameter because the default of 10 is sufficient. For files larger than 2.5 GB, you should use 10 threads as the base for the first 2.5 GB and add 1 thread for each additional 256-MB increase in file size. If you are copying a folder with a large range of file sizes, consider grouping them into similar file sizes. Having dissimilar file sizes may cause non-optimal performance. If that's not possible to group similar file sizes, you should set PerFileThreadCount based on the largest file size.

  ```
  PerFileThreadCount = 10 threads for the first 2.5 GB + 1 thread for each additional 256 MB increase in file size
  ```

  Example:

  Assuming you have 100 files ranging from 1 GB to 10 GB, we use the 10 GB as the largest file size for equation, which would read like the following.

  ```
  PerFileThreadCount = 10 + ((10 GB - 2.5 GB) / 256 MB) = 40 threads
  ```

- **Step 3: Calculate ConcurrentFilecount** - Use the total thread count and PerFileThreadCount to calculate ConcurrentFileCount based on the following equation:

  ```
  Total thread count = PerFileThreadCount * ConcurrentFileCount
  ```

  Example:

  Based on the example values we have been using

  ```
  96 = 40 * ConcurrentFileCount
  ```

  So, **ConcurrentFileCount** is **2.4**, which we can round off to **2**.

Further tuning

You might require further tuning because there is a range of file sizes to work with. The preceding calculation works well if all or most of the files are larger and closer to the 10-GB range. If instead, there are many different files sizes with many files being smaller, then you could reduce PerFileThreadCount. By reducing the

PerFileThreadCount, we can increase ConcurrentFileCount. So, if we assume that most of our files are smaller in the 5-GB range, we can redo our calculation:

```
PerFileThreadCount = 10 + ((5 GB - 2.5 GB) / 256 MB) = 20
```

So, **ConcurrentFileCount** becomes 96/20, which is 4.8, rounded off to **4**.

You can continue to tune these settings by changing the **PerFileThreadCount** up and down depending on the distribution of your file sizes.

Limitation

- **Number of files is less than ConcurrentFileCount**: If the number of files you are uploading is smaller than the **ConcurrentFileCount** that you calculated, then you should reduce **ConcurrentFileCount** to be equal to the number of files. You can use any remaining threads to increase **PerFileThreadCount**.
- **Too many threads**: If you increase thread count too much without increasing your cluster size, you run the risk of degraded performance. There can be contention issues when context-switching on the CPU.
- **Insufficient concurrency**: If the concurrency is not sufficient, then your cluster may be too small. You can increase the number of nodes in your cluster, which gives you more concurrency.
- **Throttling errors**: You may see throttling errors if your concurrency is too high. If you are seeing throttling errors, you should either reduce the concurrency or contact us.

Next steps

- Use Azure Data Lake Storage Gen1 for big data requirements
- Secure data in Data Lake Storage Gen1
- Use Azure Data Lake Analytics with Data Lake Storage Gen1
- Use Azure HDInsight with Data Lake Storage Gen1

Performance tuning guidance for Spark on HDInsight and Azure Data Lake Storage Gen1

10/3/2022 • 7 minutes to read • Edit Online

When tuning performance on Spark, you need to consider the number of apps that will be running on your cluster. By default, you can run four apps concurrently on your HDI cluster (Note: the default setting is subject to change). You may decide to use fewer apps so you can override the default settings and use more of the cluster for those apps.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1
- **Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. Make sure you enable Remote Desktop for the cluster.
- **Running Spark cluster on Data Lake Storage Gen1**. For more information, see Use HDInsight Spark cluster to analyze data in Data Lake Storage Gen1
- **Performance tuning guidelines on Data Lake Storage Gen1**. For general performance concepts, see Data Lake Storage Gen1 Performance Tuning Guidance

Parameters

When running Spark jobs, here are the most important settings that can be tuned to increase performance on Data Lake Storage Gen1:

- **Num-executors** - The number of concurrent tasks that can be executed.
- **Executor-memory** - The amount of memory allocated to each executor.
- **Executor-cores** - The number of cores allocated to each executor.

Num-executors Num-executors will set the maximum number of tasks that can run in parallel. The actual number of tasks that can run in parallel is bounded by the memory and CPU resources available in your cluster.

Executor-memory This is the amount of memory that is being allocated to each executor. The memory needed for each executor is dependent on the job. For complex operations, the memory needs to be higher. For simple operations like read and write, memory requirements will be lower. The amount of memory for each executor can be viewed in Ambari. In Ambari, navigate to Spark and view the **Configs** tab.

Executor-cores This sets the amount of cores used per executor, which determines the number of parallel threads that can be run per executor. For example, if executor-cores = 2, then each executor can run 2 parallel tasks in the executor. The executor-cores needed will be dependent on the job. I/O heavy jobs do not require a large amount of memory per task so each executor can handle more parallel tasks.

By default, two virtual YARN cores are defined for each physical core when running Spark on HDInsight. This number provides a good balance of concurrency and amount of context switching from multiple threads.

Guidance

While running Spark analytic workloads to work with data in Data Lake Storage Gen1, we recommend that you

use the most recent HDInsight version to get the best performance with Data Lake Storage Gen1. When your job is more I/O intensive, then certain parameters can be configured to improve performance. Data Lake Storage Gen1 is a highly scalable storage platform that can handle high throughput. If the job mainly consists of read or writes, then increasing concurrency for I/O to and from Data Lake Storage Gen1 could increase performance.

There are a few general ways to increase concurrency for I/O intensive jobs.

Step 1: Determine how many apps are running on your cluster – You should know how many apps are running on the cluster including the current one. The default values for each Spark setting assumes that there are 4 apps running concurrently. Therefore, you will only have 25% of the cluster available for each app. To get better performance, you can override the defaults by changing the number of executors.

Step 2: Set executor-memory – the first thing to set is the executor-memory. The memory will be dependent on the job that you are going to run. You can increase concurrency by allocating less memory per executor. If you see out of memory exceptions when you run your job, then you should increase the value for this parameter. One alternative is to get more memory by using a cluster that has higher amounts of memory or increasing the size of your cluster. More memory will enable more executors to be used, which means more concurrency.

Step 3: Set executor-cores – For I/O intensive workloads that do not have complex operations, it's good to start with a high number of executor-cores to increase the number of parallel tasks per executor. Setting executor-cores to 4 is a good start.

```
executor-cores = 4
```

Increasing the number of executor-cores will give you more parallelism so you can experiment with different executor-cores. For jobs that have more complex operations, you should reduce the number of cores per executor. If executor-cores is set higher than 4, then garbage collection may become inefficient and degrade performance.

Step 4: Determine amount of YARN memory in cluster – This information is available in Ambari. Navigate to YARN and view the Contigs tab. The YARN memory is displayed in this window. Note while you are in the window, you can also see the default YARN container size. The YARN container size is the same as memory per executor parameter.

Total YARN memory = nodes * YARN memory per node

Step 5: Calculate num-executors

Calculate memory constraint - The num-executors parameter is constrained either by memory or by CPU. The memory constraint is determined by the amount of available YARN memory for your application. Take the total YARN memory and divide that by executor-memory. The constraint needs to be de-scaled for the number of apps so we divide by the number of apps.

Memory constraint = (total YARN memory / executor memory) / # of apps

Calculate CPU constraint - The CPU constraint is calculated as the total virtual cores divided by the number of cores per executor. There are 2 virtual cores for each physical core. Similar to the memory constraint, we have divide by the number of apps.

virtual cores = (nodes in cluster * # of physical cores in node * 2) CPU constraint = (total virtual cores / # of cores per executor) / # of apps

Set num-executors – The num-executors parameter is determined by taking the minimum of the memory constraint and the CPU constraint.

num-executors = Min (total virtual Cores / # of cores per executor, available YARN memory / executor-memory)

Setting a higher number of num-executors does not necessarily increase performance. You should consider that adding more executors will add extra overhead for each additional executor, which can potentially degrade performance. Num-executors is bounded by the cluster resources.

Example Calculation

Let's say you currently have a cluster composed of 8 D4v2 nodes that is running two apps including the one you are going to run.

Step 1: Determine how many apps are running on your cluster – you know that you have two apps on your cluster, including the one you are going to run.

Step 2: Set executor-memory – for this example, we determine that 6GB of executor-memory will be sufficient for I/O intensive job.

```
executor-memory = 6GB
```

Step 3: Set executor-cores – Since this is an I/O intensive job, we can set the number of cores for each executor to four. Setting cores per executor to larger than four may cause garbage collection problems.

```
executor-cores = 4
```

Step 4: Determine amount of YARN memory in cluster – We navigate to Ambari to find out that each D4v2 has 25GB of YARN memory. Since there are 8 nodes, the available YARN memory is multiplied by 8.

Total YARN memory = nodes * YARN memory* per node Total YARN memory = 8 nodes * 25 GB = 200 GB

Step 5: Calculate num-executors – The num-executors parameter is determined by taking the minimum of the memory constraint and the CPU constraint divided by the # of apps running on Spark.

Calculate memory constraint – The memory constraint is calculated as the total YARN memory divided by the memory per executor.

Memory constraint = (total YARN memory / executor memory) / # of apps Memory constraint = (200 GB / 6 GB) / 2 Memory constraint = 16 (rounded) **Calculate CPU constraint** - The CPU constraint is calculated as the total yarn cores divided by the number of cores per executor.

YARN cores = nodes in cluster * # of cores per node * 2 YARN cores = 8 nodes * 8 cores per D14 * 2 = 128 CPU constraint = (total YARN cores / # of cores per executor) / # of apps CPU constraint = (128 / 4) / 2 CPU constraint = 16

Set num-executors

num-executors = Min (memory constraint, CPU constraint) num-executors = Min (16, 16) num-executors = 16

Performance tuning guidance for Hive on HDInsight and Azure Data Lake Storage Gen1

10/3/2022 • 3 minutes to read • Edit Online

The default settings have been set to provide good performance across many different use cases. For I/O intensive queries, Hive can be tuned to get better performance with Azure Data Lake Storage Gen1.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **A Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1
- **Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. Make sure you enable Remote Desktop for the cluster.
- **Running Hive on HDInsight**. To learn about running Hive jobs on HDInsight, see Use Hive on HDInsight
- **Performance tuning guidelines on Data Lake Storage Gen1**. For general performance concepts, see Data Lake Storage Gen1 Performance Tuning Guidance

Parameters

Here are the most important settings to tune for improved Data Lake Storage Gen1 performance:

- **hive.tez.container.size** – the amount of memory used by each tasks
- **tez.grouping.min-size** – minimum size of each mapper
- **tez.grouping.max-size** – maximum size of each mapper
- **hive.exec.reducer.bytes.per.reducer** – size of each reducer

hive.tez.container.size - The container size determines how much memory is available for each task. This is the main input for controlling the concurrency in Hive.

tez.grouping.min-size – This parameter allows you to set the minimum size of each mapper. If the number of mappers that Tez chooses is smaller than the value of this parameter, then Tez will use the value set here.

tez.grouping.max-size – The parameter allows you to set the maximum size of each mapper. If the number of mappers that Tez chooses is larger than the value of this parameter, then Tez will use the value set here.

hive.exec.reducer.bytes.per.reducer – This parameter sets the size of each reducer. By default, each reducer is 256MB.

Guidance

Set hive.exec.reducer.bytes.per.reducer – The default value works well when the data is uncompressed. For data that is compressed, you should reduce the size of the reducer.

Set hive.tez.container.size – In each node, memory is specified by yarn.nodemanager.resource.memory-mb and should be correctly set on HDI cluster by default. For additional information on setting the appropriate memory in YARN, see this post.

I/O intensive workloads can benefit from more parallelism by decreasing the Tez container size. This gives the

user more containers which increases concurrency. However, some Hive queries require a significant amount of memory (e.g. MapJoin). If the task does not have enough memory, you will get an out of memory exception during runtime. If you receive out of memory exceptions, then you should increase the memory.

The concurrent number of tasks running or parallelism will be bounded by the total YARN memory. The number of YARN containers will dictate how many concurrent tasks can run. To find the YARN memory per node, you can go to Ambari. Navigate to YARN and view the Configs tab. The YARN memory is displayed in this window.

> Total YARN memory = nodes * YARN memory per node Number of YARN containers = Total YARN memory / Tez container size

The key to improving performance using Data Lake Storage Gen1 is to increase the concurrency as much as possible. Tez automatically calculates the number of tasks that should be created so you do not need to set it.

Example Calculation

Let's say you have an 8 node D14 cluster.

> Total YARN memory = nodes * YARN memory per node Total YARN memory = 8 nodes * 96GB = 768GB Number of YARN containers = 768GB / 3072MB = 256

Limitations

Data Lake Storage Gen1 throttling

If you hit the limits of bandwidth provided by Data Lake Storage Gen1, you would start to see task failures. This could be identified by observing throttling errors in task logs. You can decrease the parallelism by increasing Tez container size. If you need more concurrency for your job, please contact us.

To check if you are getting throttled, you need to enable the debug logging on the client side. Here's how you can do that:

1. Put the following property in the log4j properties in Hive config. This can be done from Ambari view: log4j.logger.com.microsoft.azure.datalake.store=DEBUG Restart all the nodes/service for the config to take effect.
2. If you are getting throttled, you'll see the HTTP 429 error code in the hive log file. The hive log file is in /tmp/<user>/hive.log

Further information on Hive tuning

Here are a few blogs that will help tune your Hive queries:

- Optimize Hive queries for Hadoop in HDInsight
- Encoding the Hive query file in Azure HDInsight
- Ignite talk on optimize Hive on HDInsight

Performance tuning guidance for MapReduce on HDInsight and Azure Data Lake Storage Gen1

10/3/2022 • 5 minutes to read • Edit Online

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1
- **Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. Make sure you enable Remote Desktop for the cluster.
- **Using MapReduce on HDInsight**. For more information, see Use MapReduce in Hadoop on HDInsight
- **Review performance tuning guidelines for Data Lake Storage Gen1**. For general performance concepts, see Data Lake Storage Gen1 Performance Tuning Guidance

Parameters

When running MapReduce jobs, here are the most important parameters that you can configure to increase performance on Data Lake Storage Gen1:

PARAMETER	DESCRIPTION
`Mapreduce.map.memory.mb`	The amount of memory to allocate to each mapper.
`Mapreduce.job.maps`	The number of map tasks per job.
`Mapreduce.reduce.memory.mb`	The amount of memory to allocate to each reducer.
`Mapreduce.job.reduces`	The number of reduce tasks per job.

Mapreduce.map.memory / Mapreduce.reduce.memory

Adjust this number based on how much memory is needed for the map and/or reduce task. You can view the default values of `mapreduce.map.memory` and `mapreduce.reduce.memory` in Ambari via the Yarn configuration. In Ambari, navigate to YARN and view the **Configs** tab. The YARN memory will be displayed.

Mapreduce.job.maps / Mapreduce.job.reduces

This determines the maximum number of mappers or reducers to create. The number of splits determines how many mappers are created for the MapReduce job. Therefore, you may get fewer mappers than you requested if there are fewer splits than the number of mappers requested.

Guidance

Step 1: Determine number of jobs running

By default, MapReduce will use the entire cluster for your job. You can use less of the cluster by using fewer mappers than there are available containers. The guidance in this document assumes that your application is the only application running on your cluster.

Step 2: Set mapreduce.map.memory/mapreduce.reduce.memory

The size of the memory for map and reduce tasks will be dependent on your specific job. You can reduce the memory size if you want to increase concurrency. The number of concurrently running tasks depends on the number of containers. By decreasing the amount of memory per mapper or reducer, more containers can be created, which enable more mappers or reducers to run concurrently. Decreasing the amount of memory too much may cause some processes to run out of memory. If you get a heap error when running your job, increase the memory per mapper or reducer. Consider that adding more containers adds extra overhead for each additional container, which can potentially degrade performance. Another alternative is to get more memory by using a cluster that has higher amounts of memory or increasing the number of nodes in your cluster. More memory will enable more containers to be used, which means more concurrency.

Step 3: Determine total YARN memory

To tune mapreduce.job.maps/mapreduce.job.reduces, consider the amount of total YARN memory available for use. This information is available in Ambari. Navigate to YARN and view the **Configs** tab. The YARN memory is displayed in this window. Multiply the YARN memory with the number of nodes in your cluster to get the total YARN memory.

```
Total YARN memory = nodes * YARN memory per node
```

If you're using an empty cluster, then memory can be the total YARN memory for your cluster. If other applications are using memory, then you can choose to only use a portion of your cluster's memory by reducing the number of mappers or reducers to the number of containers you want to use.

Step 4: Calculate number of YARN containers

YARN containers dictate the amount of concurrency available for the job. Take total YARN memory and divide that by mapreduce.map.memory.

```
# of YARN containers = total YARN memory / mapreduce.map.memory
```

Step 5: Set mapreduce.job.maps/mapreduce.job.reduces

Set mapreduce.job.maps/mapreduce.job.reduces to at least the number of available containers. You can experiment further by increasing the number of mappers and reducers to see if you get better performance. Keep in mind that more mappers will have additional overhead so having too many mappers may degrade performance.

CPU scheduling and CPU isolation are turned off by default so the number of YARN containers is constrained by memory.

Example calculation

Let's say you currently have a cluster composed of 8 D14 nodes and you want to run an I/O intensive job. Here are the calculations you should do:

Step 1: Determine number of jobs running

For our example, we assume that our job is the only one running.

Step 2: Set mapreduce.map.memory/mapreduce.reduce.memory

For our example, you're running an I/O intensive job and decide that 3 GB of memory for map tasks is sufficient.

```
mapreduce.map.memory = 3GB
```

Step 3: Determine total YARN memory

```
total memory from the cluster is 8 nodes * 96GB of YARN memory for a D14 = 768GB
```

Step 4: Calculate # of YARN containers

```
# of YARN containers = 768 GB of available memory / 3 GB of memory = 256
```

Step 5: Set mapreduce.job.maps/mapreduce.job.reduces

```
mapreduce.map.jobs = 256
```

Limitations

Data Lake Storage Gen1 throttling

As a multi-tenant service, Data Lake Storage Gen1 sets account level bandwidth limits. If you hit these limits, you will start to see task failures. This can be identified by observing throttling errors in task logs. If you need more bandwidth for your job, please contact us.

To check if you are getting throttled, you need to enable the debug logging on the client side. Here's how you can do that:

1. Put the following property in the log4j properties in Ambari > YARN > Config > Advanced yarn-log4j: log4j.logger.com.microsoft.azure.datalake.store=DEBUG
2. Restart all the nodes/service for the config to take effect.
3. If you're getting throttled, you'll see the HTTP 429 error code in the YARN log file. The YARN log file is in /tmp/<user>/yarn.log

Examples to run

To demonstrate how MapReduce runs on Data Lake Storage Gen1, the following is some sample code that was run on a cluster with the following settings:

- 16 node D14v2
- Hadoop cluster running HDI 3.6

For a starting point, here are some example commands to run MapReduce Teragen, Terasort, and Teravalidate. You can adjust these commands based on your resources.

Teragen

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar teragen -
Dmapreduce.job.maps=2048 -Dmapreduce.map.memory.mb=3072 10000000000 adl://example/data/1TB-sort-input
```

Terasort

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar terasort -
Dmapreduce.job.maps=2048 -Dmapreduce.map.memory.mb=3072 -Dmapreduce.job.reduces=512 -
Dmapreduce.reduce.memory.mb=3072 adl://example/data/1TB-sort-input adl://example/data/1TB-sort-output
```

Teravalidate

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar teravalidate -
Dmapreduce.job.maps=512 -Dmapreduce.map.memory.mb=3072 adl://example/data/1TB-sort-output
adl://example/data/1TB-sort-validate
```

Performance tuning guidance for Storm on HDInsight and Azure Data Lake Storage Gen1

10/3/2022 • 10 minutes to read • Edit Online

Understand the factors that should be considered when you tune the performance of an Azure Storm topology. For example, it's important to understand the characteristics of the work done by the spouts and the bolts (whether the work is I/O or memory intensive). This article covers a range of performance tuning guidelines, including troubleshooting common issues.

Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1.
- **An Azure HDInsight cluster** with access to a Data Lake Storage Gen1 account. See Create an HDInsight cluster with Data Lake Storage Gen1. Make sure you enable Remote Desktop for the cluster.
- **Performance tuning guidelines on Data Lake Storage Gen1**. For general performance concepts, see Data Lake Storage Gen1 Performance Tuning Guidance.

Tune the parallelism of the topology

You might be able to improve performance by increasing the concurrency of the I/O to and from Data Lake Storage Gen1. A Storm topology has a set of configurations that determine the parallelism:

- Number of worker processes (the workers are evenly distributed across the VMs).
- Number of spout executor instances.
- Number of bolt executor instances.
- Number of spout tasks.
- Number of bolt tasks.

For example, on a cluster with 4 VMs and 4 worker processes, 32 spout executors and 32 spout tasks, and 256 bolt executors and 512 bolt tasks, consider the following:

Each supervisor, which is a worker node, has a single worker Java virtual machine (JVM) process. This JVM process manages 4 spout threads and 64 bolt threads. Within each thread, tasks are run sequentially. With the preceding configuration, each spout thread has one task, and each bolt thread has two tasks.

In Storm, here are the various components involved, and how they affect the level of parallelism you have:

- The head node (called Nimbus in Storm) is used to submit and manage jobs. These nodes have no impact on the degree of parallelism.
- The supervisor nodes. In HDInsight, this corresponds to a worker node Azure VM.
- The worker tasks are Storm processes running in the VMs. Each worker task corresponds to a JVM instance. Storm distributes the number of worker processes you specify to the worker nodes as evenly as possible.
- Spout and bolt executor instances. Each executor instance corresponds to a thread running within the workers (JVMs).
- Storm tasks. These are logical tasks that each of these threads run. This does not change the level of parallelism, so you should evaluate if you need multiple tasks per executor or not.

Get the best performance from Data Lake Storage Gen1

When working with Data Lake Storage Gen1, you get the best performance if you do the following:

- Coalesce your small appends into larger sizes (ideally 4 MB).
- Do as many concurrent requests as you can. Because each bolt thread is doing blocking reads, you want to have somewhere in the range of 8-12 threads per core. This keeps the NIC and the CPU well utilized. A larger VM enables more concurrent requests.

Example topology

Let's assume you have an eight worker node cluster with a D13v2 Azure VM. This VM has eight cores, so among the eight worker nodes, you have 64 total cores.

Let's say we do eight bolt threads per core. Given 64 cores, that means we want 512 total bolt executor instances (that is, threads). In this case, let's say we start with one JVM per VM, and mainly use the thread concurrency within the JVM to achieve concurrency. That means we need eight worker tasks (one per Azure VM), and 512 bolt executors. Given this configuration, Storm tries to distribute the workers evenly across worker nodes (also known as supervisor nodes), giving each worker node one JVM. Now within the supervisors, Storm tries to distribute the executors evenly between supervisors, giving each supervisor (that is, JVM) eight threads each.

Tune additional parameters

After you have the basic topology, you can consider whether you want to tweak any of the parameters:

- **Number of JVMs per worker node.** If you have a large data structure (for example, a lookup table) that you host in memory, each JVM requires a separate copy. Alternatively, you can use the data structure across many threads if you have fewer JVMs. For the bolt's I/O, the number of JVMs does not make as much of a difference as the number of threads added across those JVMs. For simplicity, it's a good idea to have one JVM per worker. Depending on what your bolt is doing or what application processing you require, though, you may need to change this number.
- **Number of spout executors.** Because the preceding example uses bolts for writing to Data Lake Storage Gen1, the number of spouts is not directly relevant to the bolt performance. However, depending on the amount of processing or I/O happening in the spout, it's a good idea to tune the spouts for best performance. Ensure that you have enough spouts to be able to keep the bolts busy. The output rates of the spouts should match the throughput of the bolts. The actual configuration depends on the spout.
- **Number of tasks.** Each bolt runs as a single thread. Additional tasks per bolt don't provide any additional concurrency. The only time they are of benefit is if your process of acknowledging the tuple takes a large proportion of your bolt execution time. It's a good idea to group many tuples into a larger append before you send an acknowledgment from the bolt. So, in most cases, multiple tasks provide no additional benefit.
- **Local or shuffle grouping.** When this setting is enabled, tuples are sent to bolts within the same worker process. This reduces inter-process communication and network calls. This is recommended for most topologies.

This basic scenario is a good starting point. Test with your own data to tweak the preceding parameters to achieve optimal performance.

Tune the spout

You can modify the following settings to tune the spout.

- **Tuple timeout: topology.message.timeout.secs**. This setting determines the amount of time a message takes to complete, and receive acknowledgment, before it is considered failed.
- **Max memory per worker process: worker.childopts**. This setting lets you specify additional command-line parameters to the Java workers. The most commonly used setting here is XmX, which determines the maximum memory allocated to a JVM's heap.

- **Max spout pending: topology.max.spout.pending**. This setting determines the number of tuples that can in be flight (not yet acknowledged at all nodes in the topology) per spout thread at any time.

  A good calculation to do is to estimate the size of each of your tuples. Then figure out how much memory one spout thread has. The total memory allocated to a thread, divided by this value, should give you the upper bound for the max spout pending parameter.

Tune the bolt

When you're writing to Data Lake Storage Gen1, set a size sync policy (buffer on the client side) to 4 MB. A flushing or hsync() is then performed only when the buffer size is at this value. The Data Lake Storage Gen1 driver on the worker VM automatically does this buffering, unless you explicitly perform an hsync().

The default Data Lake Storage Gen1 Storm bolt has a size sync policy parameter (fileBufferSize) that can be used to tune this parameter.

In I/O-intensive topologies, it's a good idea to have each bolt thread write to its own file, and to set a file rotation policy (fileRotationSize). When the file reaches a certain size, the stream is automatically flushed and a new file is written to. The recommended file size for rotation is 1 GB.

Handle tuple data

In Storm, a spout holds on to a tuple until it is explicitly acknowledged by the bolt. If a tuple has been read by the bolt but has not been acknowledged yet, the spout might not have persisted into Data Lake Storage Gen1 back end. After a tuple is acknowledged, the spout can be guaranteed persistence by the bolt, and can then delete the source data from whatever source it is reading from.

For best performance on Data Lake Storage Gen1, have the bolt buffer 4 MB of tuple data. Then write to the Data Lake Storage Gen1 back end as one 4 MB write. After the data has been successfully written to the store (by calling hflush()), the bolt can acknowledge the data back to the spout. This is what the example bolt supplied here does. It is also acceptable to hold a larger number of tuples before the hflush() call is made and the tuples acknowledged. However, this increases the number of tuples in flight that the spout needs to hold, and therefore increases the amount of memory required per JVM.

> **NOTE**
> Applications might have a requirement to acknowledge tuples more frequently (at data sizes less than 4 MB) for other non-performance reasons. However, that might affect the I/O throughput to the storage back end. Carefully weigh this tradeoff against the bolt's I/O performance.

If the incoming rate of tuples is not high, so the 4-MB buffer takes a long time to fill, consider mitigating this by:

- Reducing the number of bolts, so there are fewer buffers to fill.
- Having a time-based or count-based policy, where an hflush() is triggered every x flushes or every y milliseconds, and the tuples accumulated so far are acknowledged back.

The throughput in this case is lower, but with a slow rate of events, maximum throughput is not the biggest objective anyway. These mitigations help you reduce the total time that it takes for a tuple to flow through to the store. This might matter if you want a real-time pipeline even with a low event rate. Also note that if your incoming tuple rate is low, you should adjust the topology.message.timeout_secs parameter, so the tuples don't time out while they are getting buffered or processed.

Monitor your topology in Storm

While your topology is running, you can monitor it in the Storm user interface. Here are the main parameters to

look at:

- **Total process execution latency.** This is the average time one tuple takes to be emitted by the spout, processed by the bolt, and acknowledged.
- **Total bolt process latency.** This is the average time spent by the tuple at the bolt until it receives an acknowledgment.
- **Total bolt execute latency.** This is the average time spent by the bolt in the execute method.
- **Number of failures.** This refers to the number of tuples that failed to be fully processed before they timed out.
- **Capacity.** This is a measure of how busy your system is. If this number is 1, your bolts are working as fast as they can. If it is less than 1, increase the parallelism. If it is greater than 1, reduce the parallelism.

Troubleshoot common problems

Here are a few common troubleshooting scenarios.

- **Many tuples are timing out.** Look at each node in the topology to determine where the bottleneck is. The most common reason for this is that the bolts are not able to keep up with the spouts. This leads to tuples clogging the internal buffers while waiting to be processed. Consider increasing the timeout value or decreasing the max spout pending.
- **There is a high total process execution latency, but a low bolt process latency.** In this case, it is possible that the tuples are not being acknowledged fast enough. Check that there are a sufficient number of acknowledgers. Another possibility is that they are waiting in the queue for too long before the bolts start processing them. Decrease the max spout pending.
- **There is a high bolt execute latency.** This means that the execute() method of your bolt is taking too long. Optimize the code, or look at write sizes and flush behavior.

Data Lake Storage Gen1 throttling

If you hit the limits of bandwidth provided by Data Lake Storage Gen1, you might see task failures. Check task logs for throttling errors. You can decrease the parallelism by increasing container size.

To check if you are getting throttled, enable the debug logging on the client side:

1. In **Ambari** > **Storm** > **Config** > **Advanced storm-worker-log4j**, change **<root level="info">** to **<root level="debug">**. Restart all the nodes/service for the configuration to take effect.
2. Monitor the Storm topology logs on worker nodes (under /var/log/storm/worker-artifacts/<TopologyName>/<port>/worker.log) for Data Lake Storage Gen1 throttling exceptions.

Next steps

Additional performance tuning for Storm can be referenced in this blog.

For an additional example to run, see this one on GitHub.

Create HDInsight clusters with Azure Data Lake Storage Gen1 by using the Azure portal

10/3/2022 • 8 minutes to read • Edit Online

Learn how to use the Azure portal to create a HDInsight cluster with Azure Data Lake Storage Gen1 as the default storage or an additional storage. Even though additional storage is optional for a HDInsight cluster, it's recommended to store your business data in the additional storage accounts.

Prerequisites

Before you begin, ensure that you've met the following requirements:

- **An Azure subscription**. Go to Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. Follow the instructions from Get started with Azure Data Lake Storage Gen1 by using the Azure portal. You must also create a root folder on the account. In this article, a root folder called **/clusters** is used.
- **An Azure Active Directory service principal**. This how-to guide provides instructions on how to create a service principal in Azure Active Directory (Azure AD). However, to create a service principal, you must be an Azure AD administrator. If you're an administrator, you can skip this prerequisite and continue.

> **NOTE**
> You can create a service principal only if you're an Azure AD administrator. Your Azure AD administrator must create a service principal before you can create an HDInsight cluster with Data Lake Storage Gen1. Also, the service principal must be created with a certificate, as described at Create a service principal with certificate.

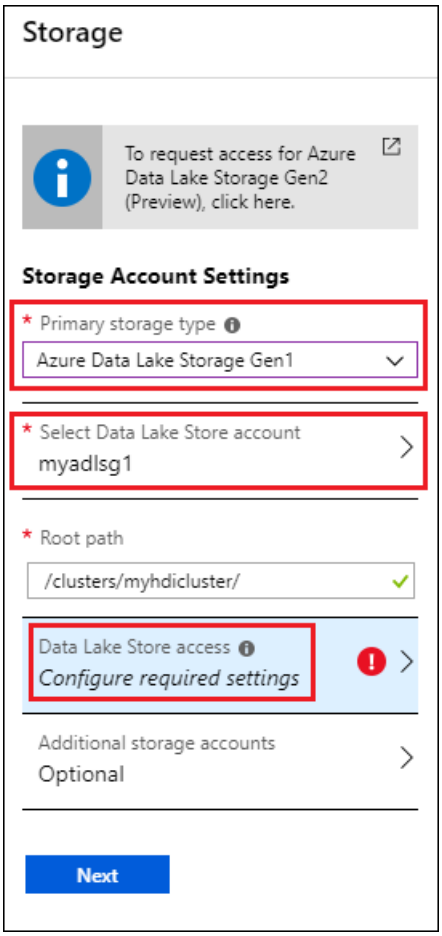
Create an HDInsight cluster

In this section, you create a HDInsight cluster with Data Lake Storage Gen1 as the default or the additional storage. This article focuses only on the part of configuring Data Lake Storage Gen1. For the general cluster creation information and procedures, see Create Hadoop clusters in HDInsight.

Create a cluster with Data Lake Storage Gen1 as default storage

To create an HDInsight cluster with a Data Lake Storage Gen1 as the default storage account:

1. Sign in to the Azure portal.
2. Follow Create clusters for the general information on creating HDInsight clusters.
3. On the **Storage** blade, under **Primary storage type**, select **Azure Data Lake Storage Gen1**, and then enter the following information:



- **Select Data Lake Store account**: Select an existing Data Lake Storage Gen1 account. An existing Data Lake Storage Gen1 account is required. See Prerequisites.
- **Root path**: Enter a path where the cluster-specific files are to be stored. On the screenshot, it is **/clusters/myhdiadlcluster/**, in which the **/clusters** folder must exist, and the Portal creates *myhdicluster* folder. The *myhdicluster* is the cluster name.
- **Data Lake Store access**: Configure access between the Data Lake Storage Gen1 account and HDInsight cluster. For instructions, see Configure Data Lake Storage Gen1 access.
- **Additional storage accounts**: Add Azure storage accounts as additional storage accounts for the cluster. To add additional Data Lake Storage Gen1 accounts is done by giving the cluster permissions on data in more Data Lake Storage Gen1 accounts while configuring a Data Lake Storage Gen1 account as the primary storage type. See Configure Data Lake Storage Gen1 access.

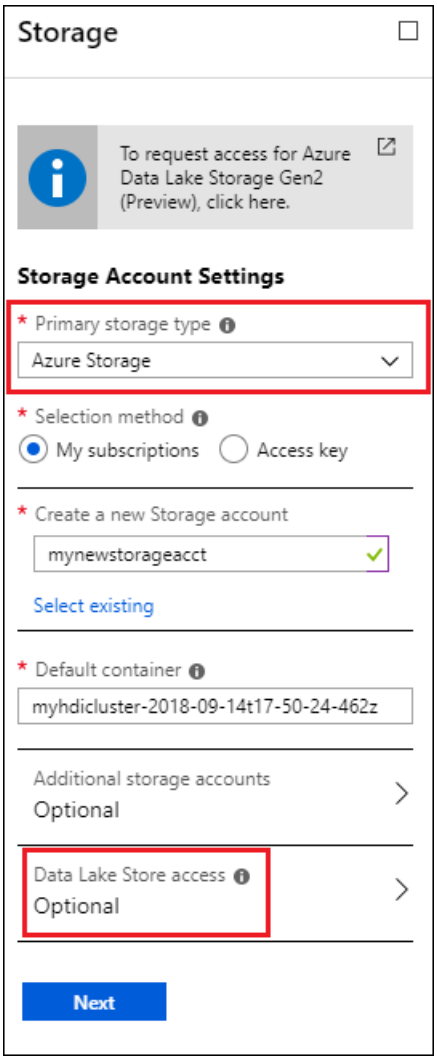
4. On the **Data Lake Store access**, click **Select**, and then continue with cluster creation as described in Create Hadoop clusters in HDInsight.

Create a cluster with Data Lake Storage Gen1 as additional storage

The following instructions create a HDInsight cluster with an Azure Blob storage account as the default storage, and a storage account with Data Lake Storage Gen1 as an additional storage.

To create a HDInsight cluster with Data Lake Storage Gen1 as an additional storage account:

1. Sign in to the Azure portal.
2. Follow Create clusters for the general information on creating HDInsight clusters.
3. On the **Storage** blade, under **Primary storage type**, select **Azure Storage**, and then enter the following information:



- **Selection method** - To specify a storage account that is part of your Azure subscription, select **My subscriptions**, and then select the storage account. To specify a storage account that is outside your Azure subscription, select **Access key**, and then provide the information for the outside storage account.
- **Default container** - Use either the default value or specify your own name.
- **Additional storage accounts** - Add more Azure storage accounts as the additional storage.
- **Data Lake Store access** - Configure access between the Data Lake Storage Gen1 account and HDInsight cluster. For instructions see Configure Data Lake Storage Gen1 access.

Configure Data Lake Storage Gen1 access

In this section, you configure Data Lake Storage Gen1 access from HDInsight clusters using an Azure Active Directory service principal.

Specify a service principal

From the Azure portal, you can either use an existing service principal or create a new one.

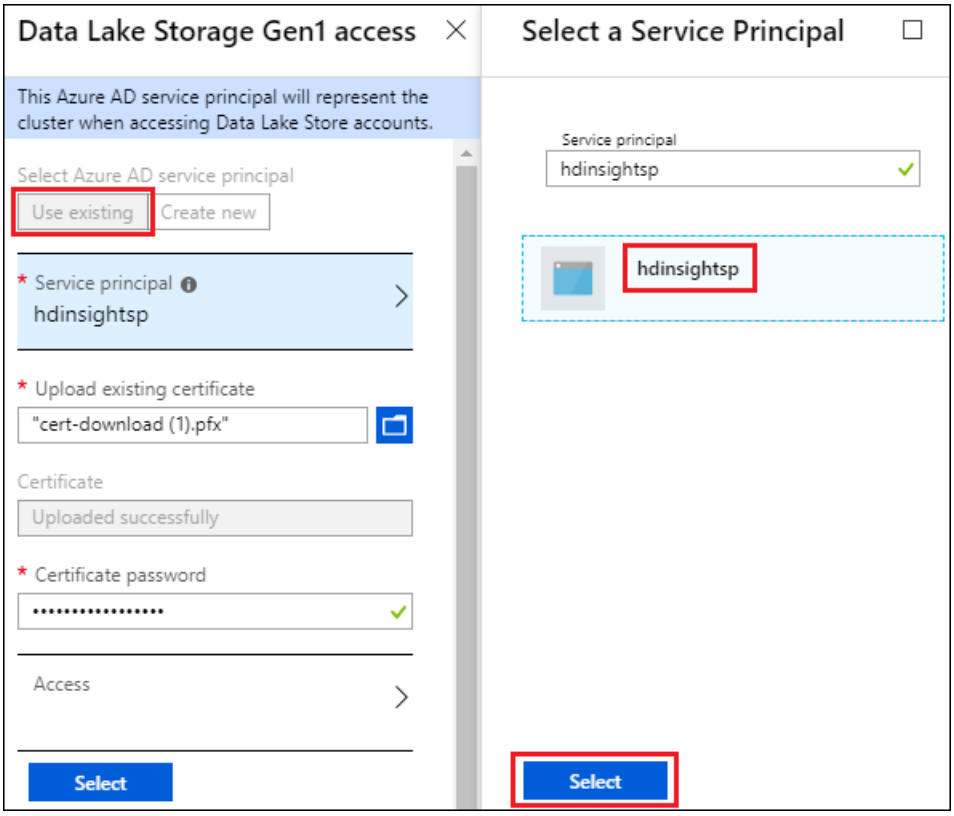
To create a service principal from the Azure portal:

1. See Create Service Principal and Certificates using Azure Active Directory.

To use an existing service principal from the Azure portal:

1. Service Principal should have owner permissions on the Storage account. See Set up permissions for the Service Principal to be owner on the storage account.

2. Select **Data Lake Store access**.
3. On the **Data Lake Storage Gen1 access** blade, select **Use existing**.
4. Select **Service principal**, and then select a service principal.
5. Upload the certificate (.pfx file) that's associated with your selected service principal, and then enter the certificate password.



6. Select **Access** to configure the folder access. See Configure file permissions.

Set up permissions for the Service Principal to be owner on the storage account

1. On the Access Control(IAM) blade of storage account click Add a role assignment.
2. On the Add a role assignment blade select Role as 'owner', and select the SPN and click save.

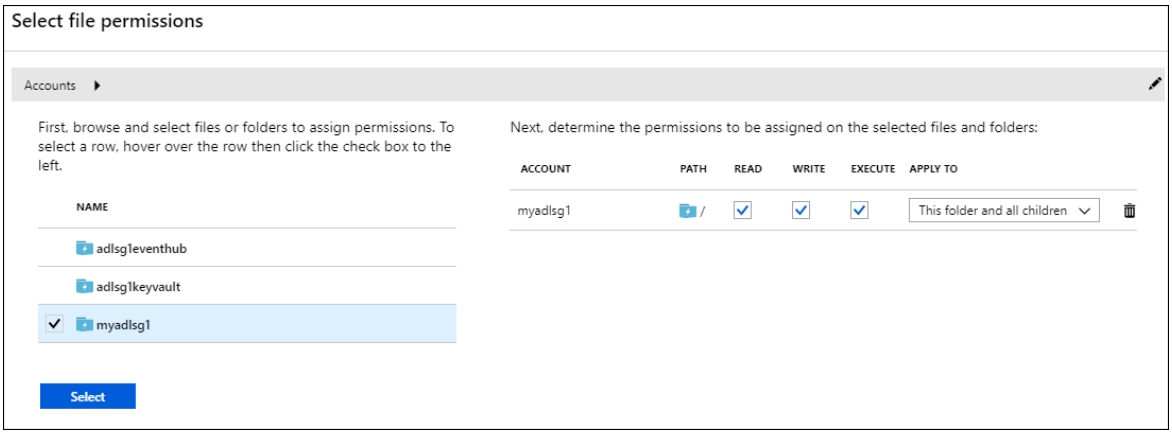
Configure file permissions

The configuration is different depending on whether the account is used as the default storage or an additional storage account:

- Used as default storage
  - permission at the root level of the Data Lake Storage Gen1 account
  - permission at the root level of the HDInsight cluster storage. For example, the **/clusters** folder used earlier in the tutorial.
- Use as an additional storage
  - Permission at the folders where you need file access.

To assign permission at the storage account with Data Lake Storage Gen1 at the root level:

1. On the **Data Lake Storage Gen1 access** blade, select **Access**. The **Select file permissions** blade is opened. It lists all the storage accounts in your subscription.
2. Hover (do not click) the mouse over the name of the account with Data Lake Storage Gen1 to make the check box visible, then select the check box.



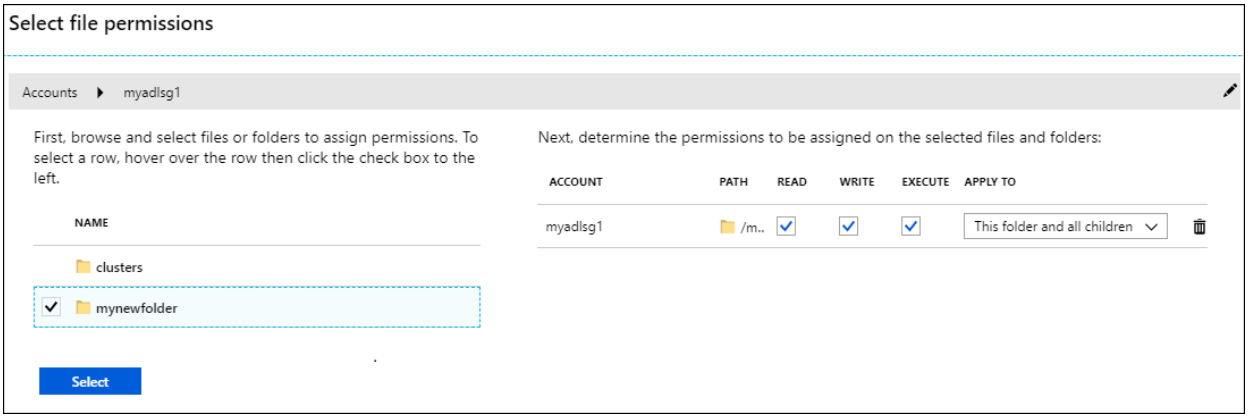
By default, **READ**, **WRITE**, AND **EXECUTE** are all selected.

3. Click **Select** on the bottom of the page.
4. Select **Run** to assign permission.
5. Select **Done**.

To assign permission at the HDInsight cluster root level:

1. On the **Data Lake Storage Gen1 access** blade, select **Access**. The **Select file permissions** blade is opened. It lists all the storage accounts with Data Lake Storage Gen1 in your subscription.
2. From the **Select file permissions** blade, select the storage account with Data Lake Storage Gen1 name to show its content.
3. Select the HDInsight cluster storage root by selecting the checkbox on the left of the folder. According to the screenshot earlier, the cluster storage root is **/clusters** folder that you specified while selecting Data Lake Storage Gen1 as default storage.
4. Set the permissions on the folder. By default, read, write, and execute are all selected.
5. Click **Select** on the bottom of the page.
6. Select **Run**.
7. Select **Done**.

If you are using Data Lake Storage Gen1 as additional storage, you must assign permission only for the folders that you want to access from the HDInsight cluster. For example, in the screenshot below, you provide access only to the **mynewfolder** folder in a storage account with Data Lake Storage Gen1.

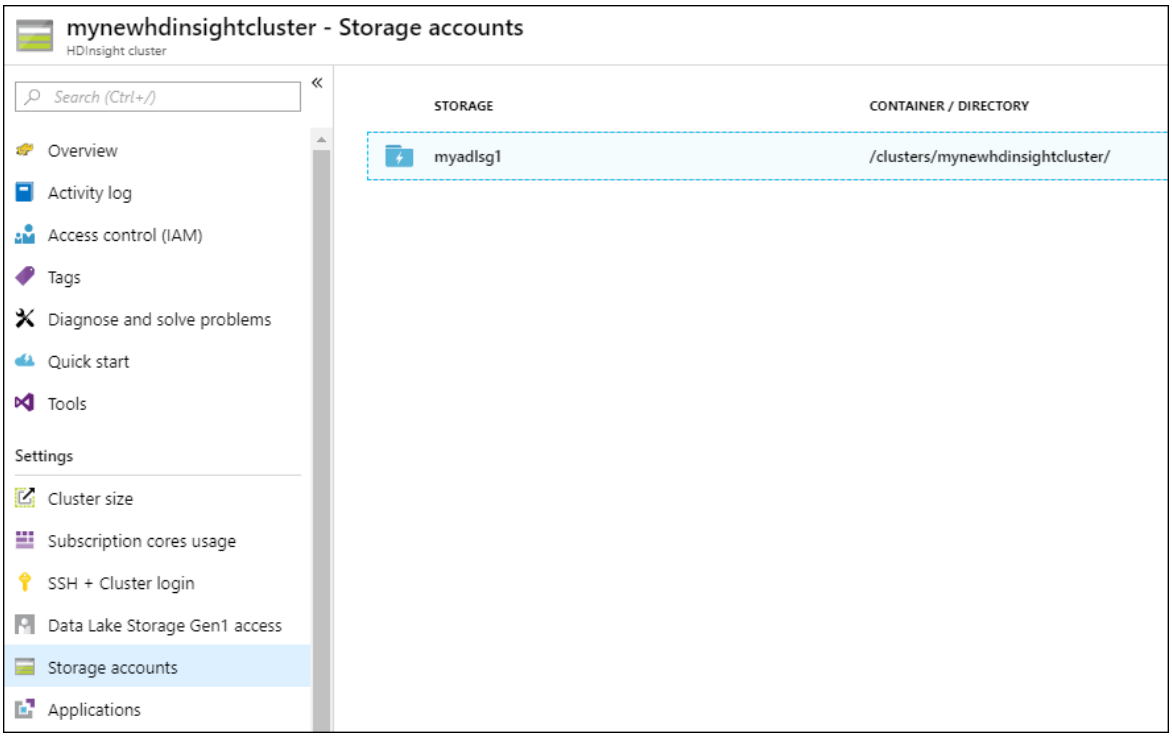


Verify cluster setup

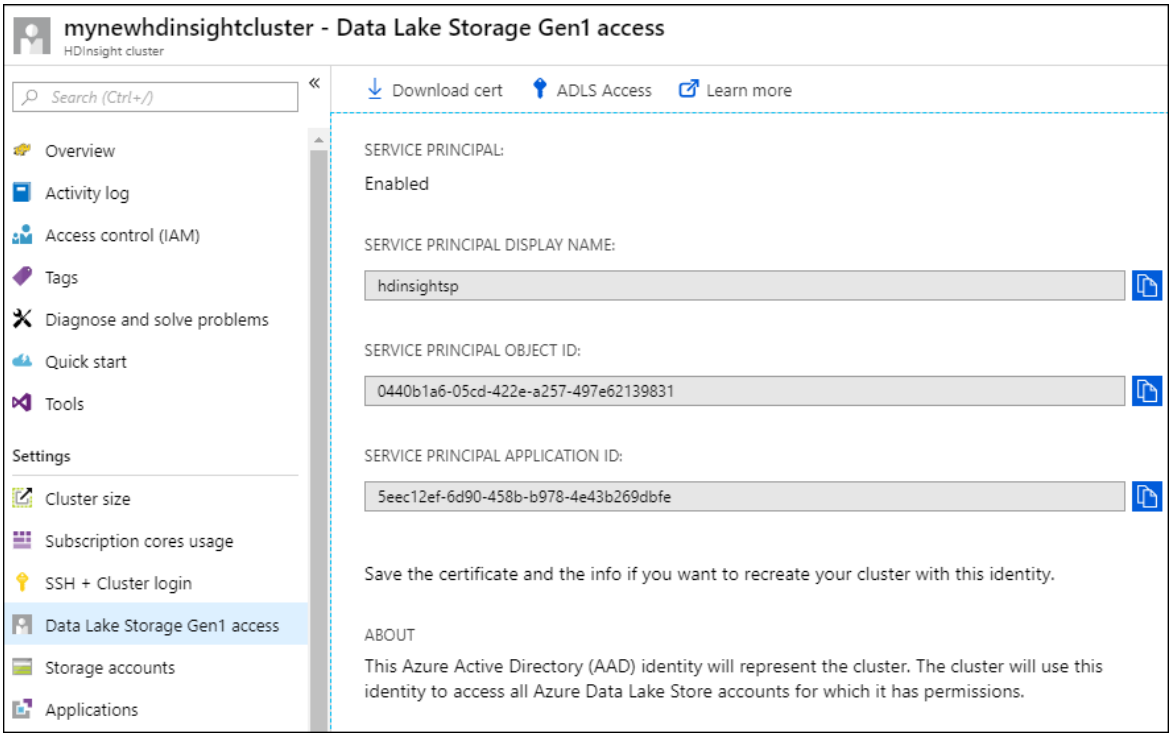
After the cluster setup is complete, on the cluster blade, verify your results by doing either or both of the following steps:

- To verify that the associated storage for the cluster is the account with Data Lake Storage Gen1 that you

specified, select **Storage accounts** in the left pane.



- To verify that the service principal is correctly associated with the HDInsight cluster, select **Data Lake Storage Gen1 access** in the left pane.



Examples

After you set up the cluster with Data Lake Storage Gen1 as your storage, see these examples of how to use HDInsight cluster to analyze the data that's stored in Data Lake Storage Gen1.

Run a Hive query against data in a Data Lake Storage Gen1 (as primary storage)

To run a Hive query, use the Hive views interface in the Ambari portal. For instructions on how to use Ambari Hive views, see Use the Hive View with Hadoop in HDInsight.

When you work with data in a Data Lake Storage Gen1, there are a few strings to change.

If you use, for example, the cluster that you created with Data Lake Storage Gen1 as primary storage, the path to the data is: *adl://<data_lake_storage_gen1_account_name>/azuredatalakestore.net/path/to/file*. A Hive query to create a table from sample data that's stored in the Data Lake Storage Gen1 looks like the following statement:

```
CREATE EXTERNAL TABLE websitelog (str string) LOCATION
'adl://hdiadlsg1storage.azuredatalakestore.net/clusters/myhdiadlcluster/HdiSamples/HdiSamples/WebsiteLogSamp
leData/SampleLog/'
```

Descriptions:

- `adl://hdiadlsg1storage.azuredatalakestore.net/` is the root of the account with Data Lake Storage Gen1.
- `/clusters/myhdiadlcluster` is the root of the cluster data that you specified while creating the cluster.
- `/HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog/` is the location of the sample file that you used in the query.

Run a Hive query against data in a Data Lake Storage Gen1 (as additional storage)

If the cluster that you created uses Blob storage as default storage, the sample data is not contained in the storage account with Data Lake Storage Gen1 that's used as additional storage. In such a case, first transfer the data from Blob storage to the storage account with Data Lake Storage Gen1, and then run the queries as shown in the preceding example.

For information on how to copy data from Blob storage to a storage account with Data Lake Storage Gen1, see the following articles:

- Use Distcp to copy data between Azure Blob storage and Data Lake Storage Gen1
- Use AdlCopy to copy data from Azure Blob storage to Data Lake Storage Gen1

Use Data Lake Storage Gen1 with a Spark cluster

You can use a Spark cluster to run Spark jobs on data that is stored in a Data Lake Storage Gen1. For more information, see Use HDInsight Spark cluster to analyze data in Data Lake Storage Gen1.

Use Data Lake Storage Gen1 in a Storm topology

See also

- Use Data Lake Storage Gen1 with Azure HDInsight clusters
- PowerShell: Create an HDInsight cluster to use Data Lake Storage Gen1

Create HDInsight clusters with Azure Data Lake Storage Gen1 as default storage by using PowerShell

10/3/2022 • 9 minutes to read • Edit Online

Learn how to use Azure PowerShell to configure Azure HDInsight clusters with Azure Data Lake Storage Gen1, as default storage. For instructions on creating an HDInsight cluster with Data Lake Storage Gen1 as additional storage, see Create an HDInsight cluster with Data Lake Storage Gen1 as additional storage.

Here are some important considerations for using HDInsight with Data Lake Storage Gen1:

- The option to create HDInsight clusters with access to Data Lake Storage Gen1 as default storage is available for HDInsight version 3.5 and 3.6.
- The option to create HDInsight clusters with access to Data Lake Storage Gen1 as default storage is *not available* for HDInsight Premium clusters.

To configure HDInsight to work with Data Lake Storage Gen1 by using PowerShell, follow the instructions in the next five sections.

Prerequisites

> **NOTE**
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Before you begin this tutorial, make sure that you meet the following requirements:

- **An Azure subscription**: Go to Get Azure free trial.
- **Azure PowerShell 1.0 or greater**: See How to install and configure PowerShell.
- **Windows Software Development Kit (SDK)**: To install Windows SDK, go to Downloads and tools for Windows 10. The SDK is used to create a security certificate.
- **Azure Active Directory service principal**: This tutorial describes how to create a service principal in Azure Active Directory (Azure AD). However, to create a service principal, you must be an Azure AD administrator. If you are an administrator, you can skip this prerequisite and proceed with the tutorial.

  > **NOTE**
  > You can create a service principal only if you are an Azure AD administrator. Your Azure AD administrator must create a service principal before you can create an HDInsight cluster with Data Lake Storage Gen1. The service principal must be created with a certificate, as described in Create a service principal with certificate.

Create an Azure Data Lake Storage Gen1 account

To create a Data Lake Storage Gen1 account, do the following:

1. From your desktop, open a PowerShell window, and then enter the snippets below. When you are prompted to sign in, sign in as one of the subscription administrators or owners.

```
# Sign in to your Azure account
Connect-AzAccount

# List all the subscriptions associated to your account
Get-AzSubscription

# Select a subscription
Set-AzContext -SubscriptionId <subscription ID>

# Register for Data Lake Storage Gen1
Register-AzResourceProvider -ProviderNamespace "Microsoft.DataLakeStore"
```

> **NOTE**
>
> If you register the Data Lake Storage Gen1 resource provider and receive an error similar to
>
> ```
> Register-AzResourceProvider : InvalidResourceNamespace: The resource namespace
> 'Microsoft.DataLakeStore' is invalid
> ```
>
> , your subscription might not be approved for Data Lake Storage Gen1. To enable your Azure subscription for Data Lake Storage Gen1, follow the instructions in Get started with Azure Data Lake Storage Gen1 by using the Azure portal.

2. A Data Lake Storage Gen1 account is associated with an Azure resource group. Start by creating a resource group.

```
$resourceGroupName = "<your new resource group name>"
New-AzResourceGroup -Name $resourceGroupName -Location "East US 2"
```

   You should see an output like this:

```
ResourceGroupName : hdiadlgrp
Location          : eastus2
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/<subscription-id>/resourceGroups/hdiadlgrp
```

3. Create a Data Lake Storage Gen1 account. The account name you specify must contain only lowercase letters and numbers.

```
$dataLakeStorageGen1Name = "<your new Data Lake Storage Gen1 name>"
New-AzDataLakeStoreAccount -ResourceGroupName $resourceGroupName -Name $dataLakeStorageGen1Name -
Location "East US 2"
```

   You should see an output like the following:

```
...
ProvisioningState           : Succeeded
State                       : Active
CreationTime                : 5/5/2017 10:53:56 PM
EncryptionState             : Enabled
...
LastModifiedTime            : 5/5/2017 10:53:56 PM
Endpoint                    : hdiadlstore.azuredatalakestore.net
DefaultGroup                :
Id                          : /subscriptions/<subscription-
id>/resourceGroups/hdiadlgrp/providers/Microsoft.DataLakeStore/accounts/hdiadlstore
Name                        : hdiadlstore
Type                        : Microsoft.DataLakeStore/accounts
Location                    : East US 2
Tags                        : {}
```

4. Using Data Lake Storage Gen1 as default storage requires you to specify a root path to which the cluster-specific files are copied during cluster creation. To create a root path, which is **/clusters/hdiadlcluster** in the snippet, use the following cmdlets:

```
$myrootdir = "/"
New-AzDataLakeStoreItem -Folder -AccountName $dataLakeStorageGen1Name -Path
$myrootdir/clusters/hdiadlcluster
```

Set up authentication for role-based access to Data Lake Storage Gen1

Every Azure subscription is associated with an Azure AD entity. Users and services that access subscription resources by using the Azure portal or the Azure Resource Manager API must first authenticate with Azure AD. Access is granted to Azure subscriptions and services by assigning them the appropriate role on an Azure resource. For services, a service principal identifies the service in Azure AD.

This section illustrates how to grant an application service, such as HDInsight, access to an Azure resource (the Data Lake Storage Gen1 account that you created earlier). You do so by creating a service principal for the application and assigning roles to it via PowerShell.

To set up Active Directory authentication for Data Lake Storage Gen1, perform the tasks in the following two sections.

Create a self-signed certificate

Make sure you have Windows SDK installed before proceeding with the steps in this section. You must have also created a directory, such as *C:\mycertdir*, where you create the certificate.

1. From the PowerShell window, go to the location where you installed Windows SDK (typically, *C:\Program Files (x86)\Windows Kits\10\bin\x86*) and use the MakeCert utility to create a self-signed certificate and a private key. Use the following commands:

```
$certificateFileDir = "<my certificate directory>"
cd $certificateFileDir

makecert -sv mykey.pvk -n "cn=HDI-ADL-SP" CertFile.cer -r -len 2048
```

   You will be prompted to enter the private key password. After the command is successfully executed, you should see **CertFile.cer** and **mykey.pvk** in the certificate directory that you specified.

2. Use the Pvk2Pfx utility to convert the .pvk and .cer files that MakeCert created to a .pfx file. Run the

following command:

```
pvk2pfx -pvk mykey.pvk -spc CertFile.cer -pfx CertFile.pfx -po <password>
```

When you are prompted, enter the private key password that you specified earlier. The value you specify for the **-po** parameter is the password that's associated with the .pfx file. After the command has been completed successfully, you should also see a **CertFile.pfx** in the certificate directory that you specified.

Create an Azure AD and a service principal

In this section, you create a service principal for an Azure AD application, assign a role to the service principal, and authenticate as the service principal by providing a certificate. To create an application in Azure AD, run the following commands:

1. Paste the following cmdlets in the PowerShell console window. Make sure that the value you specify for the **-DisplayName** property is unique. The values for **-HomePage** and **-IdentiferUris** are placeholder values and are not verified.

```
$certificateFilePath = "$certificateFileDir\CertFile.pfx"

$password = Read-Host -Prompt "Enter the password" # This is the password you specified for the .pfx
file

$certificatePFX = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2($certificateFilePath, $password)

$rawCertificateData = $certificatePFX.GetRawCertData()

$credential = [System.Convert]::ToBase64String($rawCertificateData)

$application = New-AzADApplication `
    -DisplayName "HDIADL" `
    -HomePage "https://contoso.com" `
    -IdentifierUris "https://mycontoso.com" `
    -CertValue $credential  `
    -StartDate $certificatePFX.NotBefore  `
    -EndDate $certificatePFX.NotAfter

$applicationId = $application.ApplicationId
```

2. Create a service principal by using the application ID.

```
$servicePrincipal = New-AzADServicePrincipal -ApplicationId $applicationId -Role Contributor

$objectId = $servicePrincipal.Id
```

3. Grant the service principal access to the Data Lake Storage Gen1 root and all the folders in the root path that you specified earlier. Use the following cmdlets:

```
Set-AzDataLakeStoreItemAclEntry -AccountName $dataLakeStorageGen1Name -Path / -AceType User -Id
$objectId -Permissions All
Set-AzDataLakeStoreItemAclEntry -AccountName $dataLakeStorageGen1Name -Path /clusters -AceType User -
Id $objectId -Permissions All
Set-AzDataLakeStoreItemAclEntry -AccountName $dataLakeStorageGen1Name -Path /clusters/hdiadlcluster -
AceType User -Id $objectId -Permissions All
```

Create an HDInsight Linux cluster with Data Lake Storage Gen1 as the

default storage

In this section, you create an HDInsight Hadoop Linux cluster with Data Lake Storage Gen1 as the default storage. For this release, the HDInsight cluster and Data Lake Storage Gen1 must be in the same location.

1. Retrieve the subscription tenant ID, and store it to use later.

```
$tenantID = (Get-AzContext).Tenant.TenantId
```

2. Create the HDInsight cluster by using the following cmdlets:

```
# Set these variables

$location = "East US 2"
$storageAccountName = $dataLakeStorageGen1Name    # Data Lake Storage Gen1 account name
    $storageRootPath = "<Storage root path you specified earlier>"     # e.g. /clusters/hdiadlcluster
    $clusterName = "<unique cluster name>"
$clusterNodes = <ClusterSizeInNodes>            # The number of nodes in the HDInsight cluster
$httpCredentials = Get-Credential
$sshCredentials = Get-Credential

New-AzHDInsightCluster `
       -ClusterType Hadoop `
       -OSType Linux `
       -ClusterSizeInNodes $clusterNodes `
       -ResourceGroupName $resourceGroupName `
       -ClusterName $clusterName `
       -HttpCredential $httpCredentials `
       -Location $location `
       -DefaultStorageAccountType AzureDataLakeStore `
       -DefaultStorageAccountName "$storageAccountName.azuredatalakestore.net" `
       -DefaultStorageRootPath $storageRootPath `
       -Version "3.6" `
       -SshCredential $sshCredentials `
       -AadTenantId $tenantId `
       -ObjectId $objectId `
       -CertificateFilePath $certificateFilePath `
       -CertificatePassword $password
```

   After the cmdlet has been successfully completed, you should see an output that lists the cluster details.

Run test jobs on the HDInsight cluster to use Data Lake Storage Gen1

After you have configured an HDInsight cluster, you can run test jobs on it to ensure that it can access Data Lake Storage Gen1. To do so, run a sample Hive job to create a table that uses the sample data that's already available in Data Lake Storage Gen1 at *<cluster root>/example/data/sample.log*.

In this section, you make a Secure Shell (SSH) connection into the HDInsight Linux cluster that you created, and then you run a sample Hive query.

- If you are using a Windows client to make an SSH connection into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Windows.
- If you are using a Linux client to make an SSH connection into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Linux.

1. After you have made the connection, start the Hive command-line interface (CLI) by using the following command:

```
hive
```

2. Use the CLI to enter the following statements to create a new table named **vehicles** by using the sample data in Data Lake Storage Gen1:

```
DROP TABLE log4jLogs;
    CREATE EXTERNAL TABLE log4jLogs (t1 string, t2 string, t3 string, t4 string, t5 string, t6
string, t7 string)
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
    STORED AS TEXTFILE LOCATION 'adl:///example/data/';
    SELECT t4 AS sev, COUNT(*) AS count FROM log4jLogs WHERE t4 = '[ERROR]' AND INPUT__FILE__NAME
LIKE '%.log' GROUP BY t4;
```

You should see the query output on the SSH console.

> **NOTE**
> The path to the sample data in the preceding CREATE TABLE command is `adl:///example/data/`, where `adl:///` is the cluster root. Following the example of the cluster root that's specified in this tutorial, the command is `adl://hdiadlstore.azuredatalakestore.net/clusters/hdiadlcluster`. You can either use the shorter alternative or provide the complete path to the cluster root.

Access Data Lake Storage Gen1 by using HDFS commands

After you have configured the HDInsight cluster to use Data Lake Storage Gen1, you can use Hadoop Distributed File System (HDFS) shell commands to access the store.

In this section, you make an SSH connection into the HDInsight Linux cluster that you created, and then you run the HDFS commands.

- If you are using a Windows client to make an SSH connection into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Windows.
- If you are using a Linux client to make an SSH connection into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Linux.

After you've made the connection, list the files in Data Lake Storage Gen1 by using the following HDFS file system command.

```
hdfs dfs -ls adl:///
```

You can also use the `hdfs dfs -put` command to upload some files to Data Lake Storage Gen1, and then use `hdfs dfs -ls` to verify whether the files were successfully uploaded.

See also

- Use Data Lake Storage Gen1 with Azure HDInsight clusters
- Azure portal: Create an HDInsight cluster to use Data Lake Storage Gen1

Use Azure PowerShell to create an HDInsight cluster with Azure Data Lake Storage Gen1 (as additional storage)

10/3/2022 • 10 minutes to read • Edit Online

Learn how to use Azure PowerShell to configure an HDInsight cluster with Azure Data Lake Storage Gen1, **as additional storage**. For instructions on how to create an HDInsight cluster with Data Lake Storage Gen1 as default storage, see Create an HDInsight cluster with Data Lake Storage Gen1 as default storage.

> **NOTE**
> If you are going to use Data Lake Storage Gen1 as additional storage for HDInsight cluster, we strongly recommend that you do this while you create the cluster as described in this article. Adding Data Lake Storage Gen1 as additional storage to an existing HDInsight cluster is a complicated process and prone to errors.

For supported cluster types, Data Lake Storage Gen1 can be used as a default storage or additional storage account. When Data Lake Storage Gen1 is used as additional storage, the default storage account for the clusters will still be Azure Blob storage (WASB) and the cluster-related files (such as logs, etc.) are still written to the default storage, while the data that you want to process can be stored in a Data Lake Storage Gen1. Using Data Lake Storage Gen1 as an additional storage account does not impact performance or the ability to read/write to the storage from the cluster.

Using Data Lake Storage Gen1 for HDInsight cluster storage

Here are some important considerations for using HDInsight with Data Lake Storage Gen1:

- Option to create HDInsight clusters with access to Data Lake Storage Gen1 as additional storage is available for HDInsight versions 3.2, 3.4, 3.5, and 3.6.

Configuring HDInsight to work with Data Lake Storage Gen1 using PowerShell involves the following steps:

- Create a Data Lake Storage Gen1 account
- Set up authentication for role-based access to Data Lake Storage Gen1
- Create HDInsight cluster with authentication to Data Lake Storage Gen1
- Run a test job on the cluster

Prerequisites

> **NOTE**
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Azure PowerShell 1.0 or greater**. See How to install and configure Azure PowerShell.

- **Windows SDK**. You can install it from here. You use this to create a security certificate.
- **Azure Active Directory Service Principal**. Steps in this tutorial provide instructions on how to create a service principal in Azure AD. However, you must be an Azure AD administrator to be able to create a service principal. If you are an Azure AD administrator, you can skip this prerequisite and proceed with the tutorial.

  If you are not an Azure AD administrator, you will not be able to perform the steps required to create a service principal. In such a case, your Azure AD administrator must first create a service principal before you can create an HDInsight cluster with Data Lake Storage Gen1. Also, the service principal must be created using a certificate, as described at Create a service principal with certificate.

Create a Data Lake Storage Gen1 account

Follow these steps to create a Data Lake Storage Gen1 account.

1. From your desktop, open a new Azure PowerShell window, and enter the following snippet. When prompted to log in, make sure you log in as one of the subscription administrator/owner:

   ```
   # Log in to your Azure account
   Connect-AzAccount

   # List all the subscriptions associated to your account
   Get-AzSubscription

   # Select a subscription
   Set-AzContext -SubscriptionId <subscription ID>

   # Register for Data Lake Storage Gen1
   Register-AzResourceProvider -ProviderNamespace "Microsoft.DataLakeStore"
   ```

   NOTE

   If you receive an error similar to

   ```
   Register-AzResourceProvider : InvalidResourceNamespace: The resource namespace
   'Microsoft.DataLakeStore' is invalid
   ```

   when registering the Data Lake Storage Gen1 resource provider, it is possible that your subscription is not approved for Data Lake Storage Gen1. Make sure you enable your Azure subscription for Data Lake Storage Gen1 by following these instructions.

2. A storage account with Data Lake Storage Gen1 is associated with an Azure Resource Group. Start by creating an Azure Resource Group.

   ```
   $resourceGroupName = "<your new resource group name>"
   New-AzResourceGroup -Name $resourceGroupName -Location "East US 2"
   ```

   You should see an output like this:

   ```
   ResourceGroupName : hdiadlgrp
   Location          : eastus2
   ProvisioningState : Succeeded
   Tags              :
   ResourceId        : /subscriptions/<subscription-id>/resourceGroups/hdiadlgrp
   ```

3. Create a storage account with Data Lake Storage Gen1. The account name you specify must only contain lowercase letters and numbers.

```
$dataLakeStorageGen1Name = "<your new storage account with Data Lake Storage Gen1 name>"
New-AzDataLakeStoreAccount -ResourceGroupName $resourceGroupName -Name $dataLakeStorageGen1Name -
Location "East US 2"
```

You should see an output like the following:

```
...
ProvisioningState           : Succeeded
State                       : Active
CreationTime                : 5/5/2017 10:53:56 PM
EncryptionState             : Enabled
...
LastModifiedTime            : 5/5/2017 10:53:56 PM
Endpoint                    : hdiadlstore.azuredatalakestore.net
DefaultGroup                :
Id                          : /subscriptions/<subscription-
id>/resourceGroups/hdiadlgrp/providers/Microsoft.DataLakeStore/accounts/hdiadlstore
Name                        : hdiadlstore
Type                        : Microsoft.DataLakeStore/accounts
Location                    : East US 2
Tags                        : {}
```

4. Upload some sample data to Data Lake Storage Gen1. We'll use this later in this article to verify that the data is accessible from an HDInsight cluster. If you are looking for some sample data to upload, you can get the **Ambulance Data** folder from the Azure Data Lake Git Repository.

```
$myrootdir = "/"
Import-AzDataLakeStoreItem -AccountName $dataLakeStorageGen1Name -Path "C:\<path to
data>\vehicle1_09142014.csv" -Destination $myrootdir\vehicle1_09142014.csv
```

Set up authentication for role-based access to Data Lake Storage Gen1

Every Azure subscription is associated with an Azure Active Directory. Users and services that access resources of the subscription using the Azure portal or Azure Resource Manager API must first authenticate with that Azure Active Directory. Access is granted to Azure subscriptions and services by assigning them the appropriate role on an Azure resource. For services, a service principal identifies the service in the Azure Active Directory (Azure AD). This section illustrates how to grant an application service, like HDInsight, access to an Azure resource (the storage account with Data Lake Storage Gen1 you created earlier) by creating a service principal for the application and assigning roles to that via Azure PowerShell.

To set up Active Directory authentication for Data Lake Storage Gen1, you must perform the following tasks.

- Create a self-signed certificate
- Create an application in Azure Active Directory and a Service Principal

Create a self-signed certificate

Make sure you have Windows SDK installed before proceeding with the steps in this section. You must have also created a directory, such as **C:\mycertdir**, where the certificate will be created.

1. From the PowerShell window, navigate to the location where you installed Windows SDK (typically, `C:\Program Files (x86)\Windows Kits\10\bin\x86` and use the MakeCert utility to create a self-signed certificate and a private key. Use the following commands.

```
$certificateFileDir = "<my certificate directory>"
cd $certificateFileDir

makecert -sv mykey.pvk -n "cn=HDI-ADL-SP" CertFile.cer -r -len 2048
```

You will be prompted to enter the private key password. After the command successfully executes, you should see a **CertFile.cer** and **mykey.pvk** in the certificate directory you specified.

2. Use the Pvk2Pfx utility to convert the .pvk and .cer files that MakeCert created to a .pfx file. Run the following command.

```
pvk2pfx -pvk mykey.pvk -spc CertFile.cer -pfx CertFile.pfx -po <password>
```

When prompted enter the private key password you specified earlier. The value you specify for the **-po** parameter is the password that is associated with the .pfx file. After the command successfully completes, you should also see a CertFile.pfx in the certificate directory you specified.

Create an Azure Active Directory and a service principal

In this section, you perform the steps to create a service principal for an Azure Active Directory application, assign a role to the service principal, and authenticate as the service principal by providing a certificate. Run the following commands to create an application in Azure Active Directory.

1. Paste the following cmdlets in the PowerShell console window. Make sure the value you specify for the **-DisplayName** property is unique. Also, the values for **-HomePage** and **-IdentiferUris** are placeholder values and are not verified.

```
$certificateFilePath = "$certificateFileDir\CertFile.pfx"

$password = Read-Host -Prompt "Enter the password" # This is the password you specified for the .pfx
file

$certificatePFX = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2($certificateFilePath, $password)

$rawCertificateData = $certificatePFX.GetRawCertData()

$credential = [System.Convert]::ToBase64String($rawCertificateData)

$application = New-AzADApplication `
    -DisplayName "HDIADL" `
    -HomePage "https://contoso.com" `
    -IdentifierUris "https://mycontoso.com" `
    -CertValue $credential  `
    -StartDate $certificatePFX.NotBefore  `
    -EndDate $certificatePFX.NotAfter

$applicationId = $application.ApplicationId
```

2. Create a service principal using the application ID.

```
$servicePrincipal = New-AzADServicePrincipal -ApplicationId $applicationId -Role Contributor

 $objectId = $servicePrincipal.Id
```

3. Grant the service principal access to the Data Lake Storage Gen1 folder and the file that you will access from the HDInsight cluster. The snippet below provides access to the root of the storage account with Data Lake Storage Gen1 (where you copied the sample data file), and the file itself.

```
Set-AzDataLakeStoreItemAclEntry -AccountName $dataLakeStorageGen1Name -Path / -AceType User -Id
$objectId -Permissions All
Set-AzDataLakeStoreItemAclEntry -AccountName $dataLakeStorageGen1Name -Path /vehicle1_09142014.csv -
AceType User -Id $objectId -Permissions All
```

Create an HDInsight Linux cluster with Data Lake Storage Gen1 as additional storage

In this section, we create an HDInsight Hadoop Linux cluster with Data Lake Storage Gen1 as additional storage. For this release, the HDInsight cluster and storage account with Data Lake Storage Gen1 must be in the same location.

1. Start with retrieving the subscription tenant ID. You will need that later.

   ```
   $tenantID = (Get-AzContext).Tenant.TenantId
   ```

2. For this release, for a Hadoop cluster, Data Lake Storage Gen1 can only be used as an additional storage for the cluster. The default storage will still be the Azure Blob storage (WASB). So, we'll first create the storage account and storage containers required for the cluster.

   ```
   # Create an Azure storage account
   $location = "East US 2"
   $storageAccountName = "<StorageAccountName>"   # Provide a Storage account name

   New-AzStorageAccount -ResourceGroupName $resourceGroupName -StorageAccountName $storageAccountName -
   Location $location -Type Standard_GRS

   # Create an Azure Blob Storage container
   $containerName = "<ContainerName>"              # Provide a container name
   $storageAccountKey = (Get-AzStorageAccountKey -Name $storageAccountName -ResourceGroupName
   $resourceGroupName)[0].Value
   $destContext = New-AzStorageContext -StorageAccountName $storageAccountName -StorageAccountKey
   $storageAccountKey
   New-AzStorageContainer -Name $containerName -Context $destContext
   ```

3. Create the HDInsight cluster. Use the following cmdlets.

   ```
   # Set these variables
   $clusterName = $containerName                   # As a best practice, have the same name for the
   cluster and container
   $clusterNodes = <ClusterSizeInNodes>            # The number of nodes in the HDInsight cluster
   $httpCredentials = Get-Credential
   $sshCredentials = Get-Credential

   New-AzHDInsightCluster -ClusterName $clusterName -ResourceGroupName $resourceGroupName -
   HttpCredential $httpCredentials -Location $location -DefaultStorageAccountName
   "$storageAccountName.blob.core.windows.net" -DefaultStorageAccountKey $storageAccountKey -
   DefaultStorageContainer $containerName  -ClusterSizeInNodes $clusterNodes -ClusterType Hadoop -
   Version "3.4" -OSType Linux -SshCredential $sshCredentials -ObjectID $objectId -AadTenantId $tenantID
   -CertificateFilePath $certificateFilePath -CertificatePassword $password
   ```

   After the cmdlet successfully completes, you should see an output listing the cluster details.

Run test jobs on the HDInsight cluster to use the Data Lake Storage Gen1

After you have configured an HDInsight cluster, you can run test jobs on the cluster to test that the HDInsight cluster can access Data Lake Storage Gen1. To do so, we will run a sample Hive job that creates a table using the sample data that you uploaded earlier to your storage account with Data Lake Storage Gen1.

In this section you will SSH into the HDInsight Linux cluster you created and run the a sample Hive query.

- If you are using a Windows client to SSH into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Windows.
- If you are using a Linux client to SSH into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Linux

1. Once connected, start the Hive CLI by using the following command:

   ```
   hive
   ```

2. Using the CLI, enter the following statements to create a new table named **vehicles** by using the sample data in Data Lake Storage Gen1:

   ```
   DROP TABLE vehicles;
   CREATE EXTERNAL TABLE vehicles (str string) LOCATION
   'adl://<mydatalakestoragegen1>.azuredatalakestore.net:443/';
   SELECT * FROM vehicles LIMIT 10;
   ```

   You should see an output similar to the following:

   ```
   1,1,2014-09-14 00:00:03,46.81006,-92.08174,51,S,1
   1,2,2014-09-14 00:00:06,46.81006,-92.08174,13,NE,1
   1,3,2014-09-14 00:00:09,46.81006,-92.08174,48,NE,1
   1,4,2014-09-14 00:00:12,46.81006,-92.08174,30,W,1
   1,5,2014-09-14 00:00:15,46.81006,-92.08174,47,S,1
   1,6,2014-09-14 00:00:18,46.81006,-92.08174,9,S,1
   1,7,2014-09-14 00:00:21,46.81006,-92.08174,53,N,1
   1,8,2014-09-14 00:00:24,46.81006,-92.08174,63,SW,1
   1,9,2014-09-14 00:00:27,46.81006,-92.08174,4,NE,1
   1,10,2014-09-14 00:00:30,46.81006,-92.08174,31,N,1
   ```

Access Data Lake Storage Gen1 using HDFS commands

Once you have configured the HDInsight cluster to use Data Lake Storage Gen1, you can use the HDFS shell commands to access the store.

In this section you will SSH into the HDInsight Linux cluster you created and run the HDFS commands.

- If you are using a Windows client to SSH into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Windows.
- If you are using a Linux client to SSH into the cluster, see Use SSH with Linux-based Hadoop on HDInsight from Linux

Once connected, use the following HDFS filesystem command to list the files in the storage account with Data Lake Storage Gen1.

```
hdfs dfs -ls adl://<storage account with Data Lake Storage Gen1 name>.azuredatalakestore.net:443/
```

This should list the file that you uploaded earlier to Data Lake Storage Gen1.

```
15/09/17 21:41:15 INFO web.CaboWebHdfsFileSystem: Replacing original urlConnectionFactory with
org.apache.hadoop.hdfs.web.URLConnectionFactory@21a728d6
Found 1 items
-rwxrwxrwx   0 NotSupportYet NotSupportYet     671388 2015-09-16 22:16
adl://mydatalakestoragegen1.azuredatalakestore.net:443/mynewfolder
```

You can also use the `hdfs dfs -put` command to upload some files to Data Lake Storage Gen1, and then use `hdfs dfs -ls` to verify whether the files were successfully uploaded.

See Also

- Use Data Lake Storage Gen1 with Azure HDInsight clusters
- Portal: Create an HDInsight cluster to use Data Lake Storage Gen1

Create an HDInsight cluster with Azure Data Lake Storage Gen1 using Azure Resource Manager template

10/3/2022 • 6 minutes to read • Edit Online

Learn how to use Azure PowerShell to configure an HDInsight cluster with Azure Data Lake Storage Gen1, **as additional storage**.

For supported cluster types, Data Lake Storage Gen1 can be used as a default storage or as an additional storage account. When Data Lake Storage Gen1 is used as additional storage, the default storage account for the clusters will still be Azure Blob storage (WASB) and the cluster-related files (such as logs, etc.) are still written to the default storage, while the data that you want to process can be stored in a Data Lake Storage Gen1 account. Using Data Lake Storage Gen1 as an additional storage account does not impact performance or the ability to read/write to the storage from the cluster.

Using Data Lake Storage Gen1 for HDInsight cluster storage

Here are some important considerations for using HDInsight with Data Lake Storage Gen1:

- Option to create HDInsight clusters with access to Data Lake Storage Gen1 as default storage is available for HDInsight version 3.5 and 3.6.
- Option to create HDInsight clusters with access to Data Lake Storage Gen1 as additional storage is available for HDInsight versions 3.2, 3.4, 3.5, and 3.6.

In this article, we provision a Hadoop cluster with Data Lake Storage Gen1 as additional storage. For instructions on how to create a Hadoop cluster with Data Lake Storage Gen1 as default storage, see Create an HDInsight cluster with Data Lake Storage Gen1 using Azure portal.

Prerequisites

> **NOTE**
> To interact with Azure, the Azure Az PowerShell module is recommended. See Install Azure PowerShell to get started. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Azure PowerShell 1.0 or greater**. See How to install and configure Azure PowerShell.
- **Azure Active Directory Service Principal**. Steps in this tutorial provide instructions on how to create a service principal in Azure AD. However, you must be an Azure AD administrator to be able to create a service principal. If you are an Azure AD administrator, you can skip this prerequisite and proceed with the tutorial.

  If you are not an Azure AD administrator, you will not be able to perform the steps required to create a service principal. In such a case, your Azure AD administrator must first create a service principal before you can create an HDInsight cluster with Data Lake Storage Gen1. Also, the service principal must

be created using a certificate, as described at Create a service principal with certificate.

Create an HDInsight cluster with Data Lake Storage Gen1

The Resource Manager template, and the prerequisites for using the template, are available on GitHub at Deploy a HDInsight Linux cluster with new Data Lake Storage Gen1. Follow the instructions provided at this link to create an HDInsight cluster with Data Lake Storage Gen1 as the additional storage.

The instructions at the link mentioned above require PowerShell. Before you start with those instructions, make sure you log in to your Azure account. From your desktop, open a new Azure PowerShell window, and enter the following snippets. When prompted to log in, make sure you log in as one of the subscription administrators/owner:

```
# Log in to your Azure account
Connect-AzAccount

# List all the subscriptions associated to your account
Get-AzSubscription

# Select a subscription
Set-AzContext -SubscriptionId <subscription ID>
```

The template deploys these resource types:

- Microsoft.DataLakeStore/accounts
- Microsoft.Storage/storageAccounts
- Microsoft.HDInsight/clusters

Upload sample data to Data Lake Storage Gen1

The Resource Manager template creates a new storage account with Data Lake Storage Gen1 and associates it with the HDInsight cluster. You must now upload some sample data to Data Lake Storage Gen1. You'll need this data later in the tutorial to run jobs from an HDInsight cluster that access data in the storage account with Data Lake Storage Gen1. For instructions on how to upload data, see Upload a file to Data Lake Storage Gen1. If you are looking for some sample data to upload, you can get the **Ambulance Data** folder from the Azure Data Lake Git Repository.

Set relevant ACLs on the sample data

To make sure the sample data you upload is accessible from the HDInsight cluster, you must ensure that the Azure AD application that is used to establish identity between the HDInsight cluster and Data Lake Storage Gen1 has access to the file/folder you are trying to access. To do this, perform the following steps.

1. Find the name of the Azure AD application that is associated with HDInsight cluster and the storage account with Data Lake Storage Gen1. One way to look for the name is to open the HDInsight cluster blade that you created using the Resource Manager template, click the **Cluster Azure AD Identity** tab, and look for the value of **Service Principal Display Name**.
2. Now, provide access to this Azure AD application on the file/folder that you want to access from the HDInsight cluster. To set the right ACLs on the file/folder in Data Lake Storage Gen1, see Securing data in Data Lake Storage Gen1.

Run test jobs on the HDInsight cluster to use Data Lake Storage Gen1

After you have configured an HDInsight cluster, you can run test jobs on the cluster to test that the HDInsight cluster can access Data Lake Storage Gen1. To do so, we will run a sample Hive job that creates a table using the

sample data that you uploaded earlier to your storage account with Data Lake Storage Gen1.

In this section, you SSH into an HDInsight Linux cluster and run the sample Hive query. If you are using a Windows client, we recommend using **PuTTY**, which can be downloaded from https://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

For more information on using PuTTY, see Use SSH with Linux-based Hadoop on HDInsight from Windows.

1. Once connected, start the Hive CLI by using the following command:

   ```
   hive
   ```

2. Using the CLI, enter the following statements to create a new table named **vehicles** by using the sample data in Data Lake Storage Gen1:

   ```
   DROP TABLE vehicles;
   CREATE EXTERNAL TABLE vehicles (str string) LOCATION
   'adl://<mydatalakestoragegen1>.azuredatalakestore.net:443/';
   SELECT * FROM vehicles LIMIT 10;
   ```

   You should see output similar to the following:

   ```
   1,1,2014-09-14 00:00:03,46.81006,-92.08174,51,S,1
   1,2,2014-09-14 00:00:06,46.81006,-92.08174,13,NE,1
   1,3,2014-09-14 00:00:09,46.81006,-92.08174,48,NE,1
   1,4,2014-09-14 00:00:12,46.81006,-92.08174,30,W,1
   1,5,2014-09-14 00:00:15,46.81006,-92.08174,47,S,1
   1,6,2014-09-14 00:00:18,46.81006,-92.08174,9,S,1
   1,7,2014-09-14 00:00:21,46.81006,-92.08174,53,N,1
   1,8,2014-09-14 00:00:24,46.81006,-92.08174,63,SW,1
   1,9,2014-09-14 00:00:27,46.81006,-92.08174,4,NE,1
   1,10,2014-09-14 00:00:30,46.81006,-92.08174,31,N,1
   ```

Access Data Lake Storage Gen1 using HDFS commands

Once you have configured the HDInsight cluster to use Data Lake Storage Gen1, you can use the HDFS shell commands to access the store.

In this section, you SSH into an HDInsight Linux cluster and run the HDFS commands. If you are using a Windows client, we recommend using **PuTTY**, which can be downloaded from https://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

For more information on using PuTTY, see Use SSH with Linux-based Hadoop on HDInsight from Windows.

Once connected, use the following HDFS filesystem command to list the files in the storage account with Data Lake Storage Gen1.

```
hdfs dfs -ls adl://<storage account with Data Lake Storage Gen1 name>.azuredatalakestore.net:443/
```

This should list the file that you uploaded earlier to Data Lake Storage Gen1.

```
15/09/17 21:41:15 INFO web.CaboWebHdfsFileSystem: Replacing original urlConnectionFactory with
org.apache.hadoop.hdfs.web.URLConnectionFactory@21a728d6
Found 1 items
-rwxrwxrwx   0 NotSupportYet NotSupportYet     671388 2015-09-16 22:16
adl://mydatalakestoragegen1.azuredatalakestore.net:443/mynewfolder
```

You can also use the `hdfs dfs -put` command to upload some files to Data Lake Storage Gen1, and then use `hdfs dfs -ls` to verify whether the files were successfully uploaded.

Next steps

- Copy data from Azure Storage Blobs to Data Lake Storage Gen1
- Use Data Lake Storage Gen1 with Azure HDInsight clusters

Access Azure Data Lake Storage Gen1 from VMs within an Azure VNET

10/3/2022 • 2 minutes to read • Edit Online

Azure Data Lake Storage Gen1 is a PaaS service that runs on public Internet IP addresses. Any server that can connect to the public Internet can typically connect to Azure Data Lake Storage Gen1 endpoints as well. By default, all VMs that are in Azure VNETs can access the Internet and hence can access Azure Data Lake Storage Gen1. However, it is possible to configure VMs in a VNET to not have access to the Internet. For such VMs, access to Azure Data Lake Storage Gen1 is restricted as well. Blocking public Internet access for VMs in Azure VNETs can be done using any of the following approaches:

- By configuring Network Security Groups (NSG)
- By configuring User Defined Routes (UDR)
- By exchanging routes via BGP (industry standard dynamic routing protocol), when ExpressRoute is used, that block access to the Internet

In this article, you will learn how to enable access to Azure Data Lake Storage Gen1 from Azure VMs, which have been restricted to access resources using one of the three methods listed previously.

Enabling connectivity to Azure Data Lake Storage Gen1 from VMs with restricted connectivity

To access Azure Data Lake Storage Gen1 from such VMs, you must configure them to access the IP address for the region where the Azure Data Lake Storage Gen1 account is available. You can identify the IP addresses for your Data Lake Storage Gen1 account regions by resolving the DNS names of your accounts (`<account>.azuredatalakestore.net`). To resolve DNS names of your accounts, you can use tools such as **nslookup**. Open a command prompt on your computer and run the following command:

```
nslookup mydatastore.azuredatalakestore.net
```

The output resembles the following. The value against **Address** property is the IP address associated with your Data Lake Storage Gen1 account.

```
Non-authoritative answer:
Name:    1434ceb1-3a4b-4bc0-9c69-a0823fd69bba-mydatastore.projectcabostore.net
Address:  104.44.88.112
Aliases:  mydatastore.azuredatalakestore.net
```

Enabling connectivity from VMs restricted by using NSG

When an NSG rule is used to block access to the Internet, then you can create another NSG that allows access to the Data Lake Storage Gen1 IP Address. For more information about NSG rules, see Network security groups overview. For instructions on how to create NSGs, see How to create a network security group.

Enabling connectivity from VMs restricted by using UDR or ExpressRoute

When routes, either UDRs or BGP-exchanged routes, are used to block access to the Internet, a special route needs to be configured so that VMs in such subnets can access Data Lake Storage Gen1 endpoints. For more information, see User-defined routes overview. For instructions on creating UDRs, see Create UDRs in Resource Manager.

Enabling connectivity from VMs restricted by using ExpressRoute

When an ExpressRoute circuit is configured, the on-premises servers can access Data Lake Storage Gen1 through public peering. More details on configuring ExpressRoute for public peering is available at ExpressRoute FAQs.

See also

- Overview of Azure Data Lake Storage Gen1
- Securing data stored in Azure Data Lake Storage Gen1

Get started with Azure Data Lake Analytics using the Azure portal

10/3/2022 • 2 minutes to read • Edit Online

This article describes how to use the Azure portal to create Azure Data Lake Analytics accounts, define jobs in U-SQL, and submit jobs to the Data Lake Analytics service.

Prerequisites

Before you begin this tutorial, you must have an **Azure subscription**. See Get Azure free trial.

Create a Data Lake Analytics account

Now, you will create a Data Lake Analytics and an Azure Data Lake Storage Gen1 account at the same time. This step is simple and only takes about 60 seconds to finish.

1. Sign on to the Azure portal.
2. Click **Create a resource** > **Data + Analytics** > **Data Lake Analytics**.
3. Select values for the following items:
   - **Name**: Name your Data Lake Analytics account (Only lower case letters and numbers allowed).
   - **Subscription**: Choose the Azure subscription used for the Analytics account.
   - **Resource Group**. Select an existing Azure Resource Group or create a new one.
   - **Location**. Select an Azure data center for the Data Lake Analytics account.
   - **Data Lake Storage Gen1**: Follow the instruction to create a new Data Lake Storage Gen1 account, or select an existing one.
4. Optionally, select a pricing tier for your Data Lake Analytics account.
5. Click **Create**.

Your first U-SQL script

The following text is a very simple U-SQL script. All it does is define a small dataset within the script and then write that dataset out to the default Data Lake Storage Gen1 account as a file called `/data.csv`.

```
@a  =
    SELECT * FROM
        (VALUES
            ("Contoso", 1500.0),
            ("Woodgrove", 2700.0)
        ) AS
              D( customer, amount );
OUTPUT @a
    TO "/data.csv"
    USING Outputters.Csv();
```

Submit a U-SQL job

1. From the Data Lake Analytics account, select **New Job**.
2. Paste in the text of the preceding U-SQL script. Name the job.
3. Select **Submit** button to start the job.

4. Monitor the **Status** of the job, and wait until the job status changes to **Succeeded**.
5. Select the **Data** tab, then select the **Outputs** tab. Select the output file named `data.csv` and view the output data.

See also

- To get started developing U-SQL applications, see Develop U-SQL scripts using Data Lake Tools for Visual Studio.
- To learn U-SQL, see Get started with Azure Data Lake Analytics U-SQL language.
- For management tasks, see Manage Azure Data Lake Analytics using Azure portal.

Use Azure Data Lake Storage Gen1 to capture data from Event Hubs

10/3/2022 • 3 minutes to read • Edit Online

Learn how to use Azure Data Lake Storage Gen1 to capture data received by Azure Event Hubs.

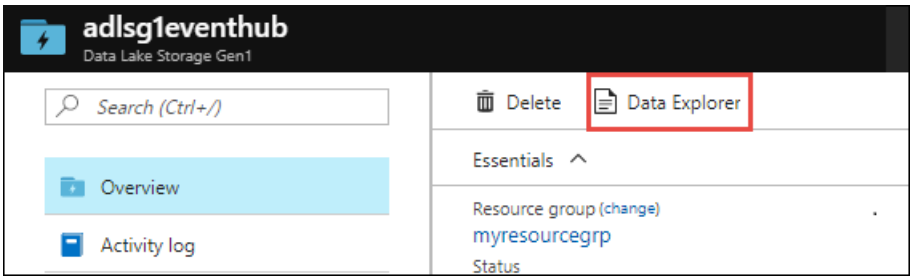
Prerequisites

- **An Azure subscription**. See Get Azure free trial.
- **An Azure Data Lake Storage Gen1 account**. For instructions on how to create one, see Get started with Azure Data Lake Storage Gen1.
- **An Event Hubs namespace**. For instructions, see Create an Event Hubs namespace. Make sure the Data Lake Storage Gen1 account and the Event Hubs namespace are in the same Azure subscription.

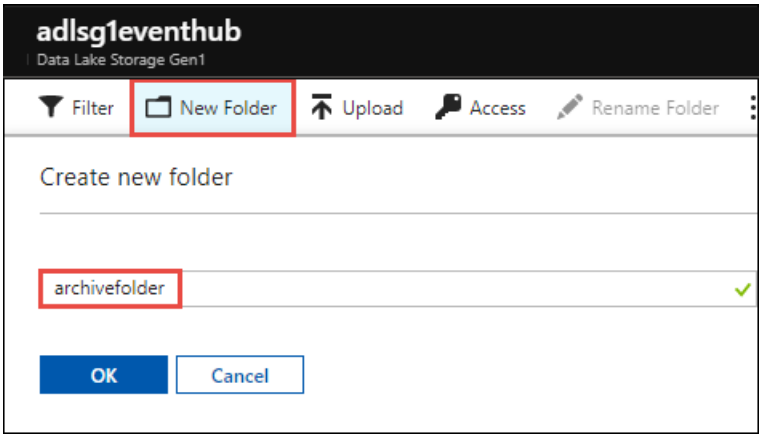
Assign permissions to Event Hubs

In this section, you create a folder within the account where you want to capture the data from Event Hubs. You also assign permissions to Event Hubs so that it can write data into a Data Lake Storage Gen1 account.

1. Open the Data Lake Storage Gen1 account where you want to capture data from Event Hubs and then click on **Data Explorer**.

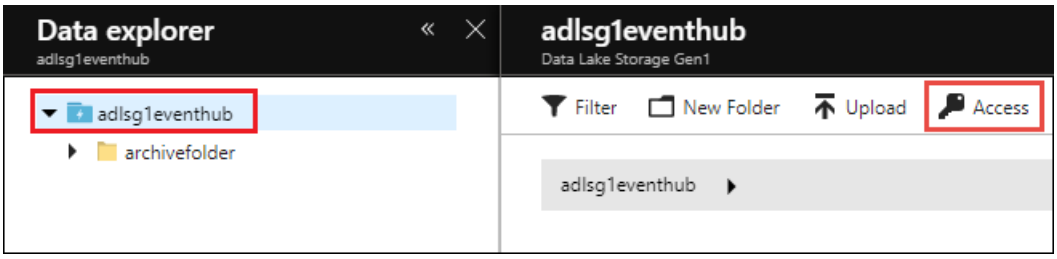


2. Click **New Folder** and then enter a name for folder where you want to capture the data.

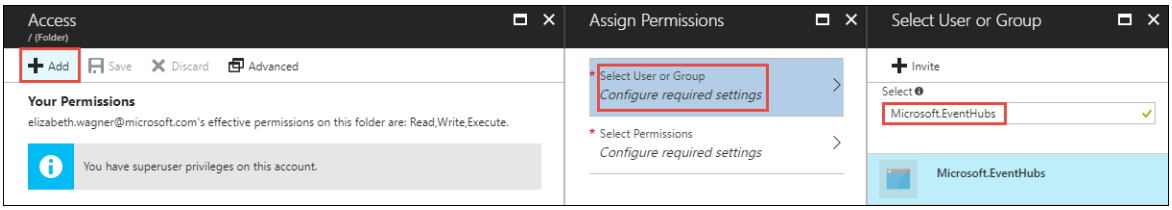


3. Assign permissions at the root of Data Lake Storage Gen1.

   a. Click **Data Explorer**, select the root of the Data Lake Storage Gen1 account, and then click **Access**.



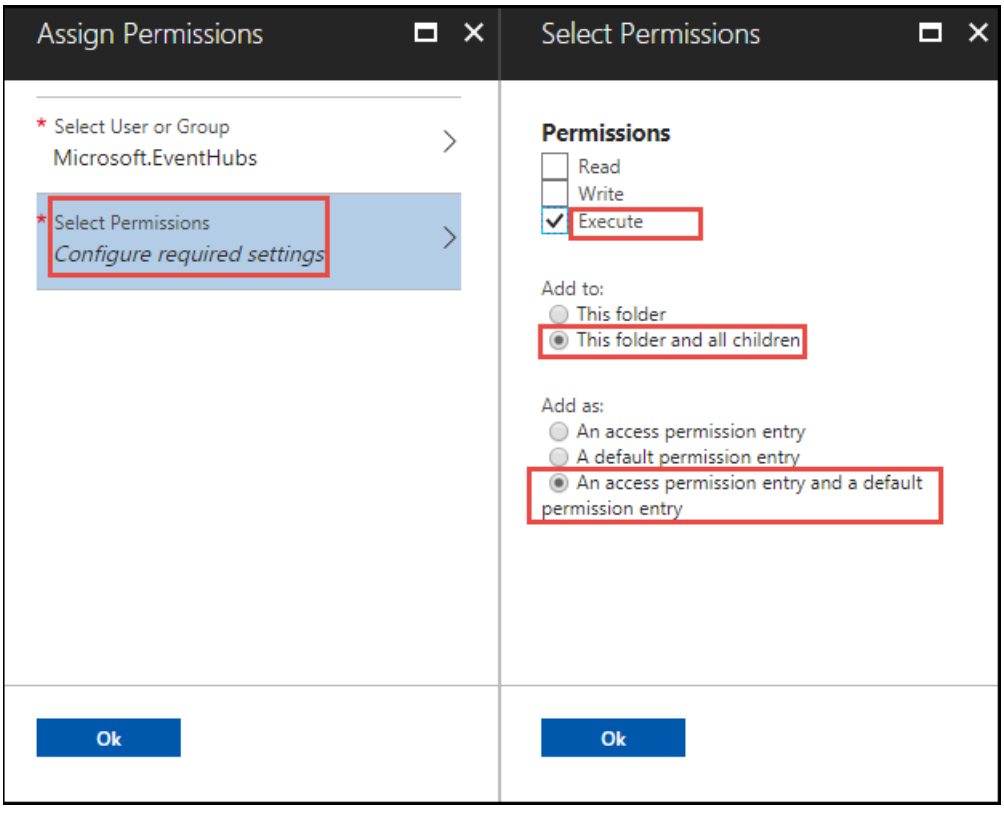
b. Under **Access**, click **Add**, click **Select User or Group**, and then search for `Microsoft.EventHubs`.



Click **Select**.

c. Under **Assign Permissions**, click **Select Permissions**. Set **Permissions** to **Execute**. Set **Add to** to **This folder and all children**. Set **Add as** to **An access permission entry and a default permission entry**.

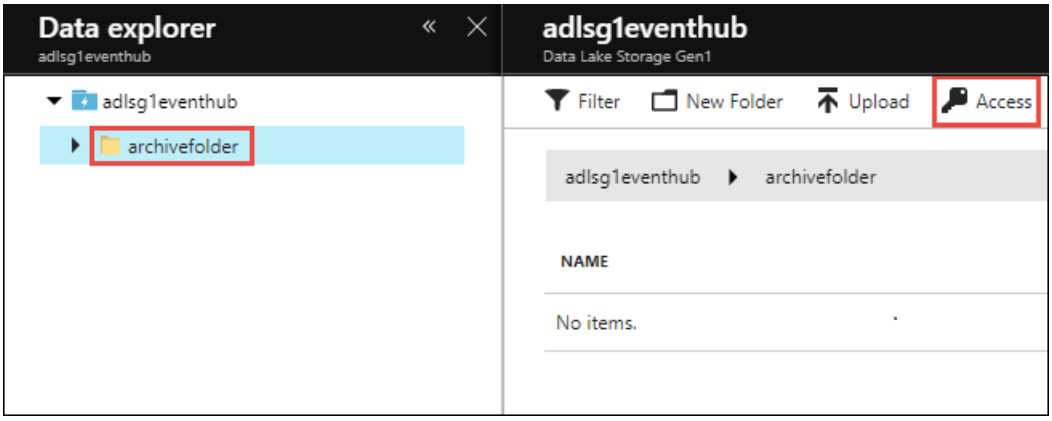
> **IMPORTANT**
>
> When creating a new folder hierarchy for capturing data received by Azure Event Hubs, this is an easy way to ensure access to the destination folder. However, adding permissions to all children of a top level folder with many child files and folders may take a long time. If your root folder contains a large number of files and folders, it may be faster to add **Execute** permissions for `Microsoft.EventHubs` individually to each folder in the path to your final destination folder.



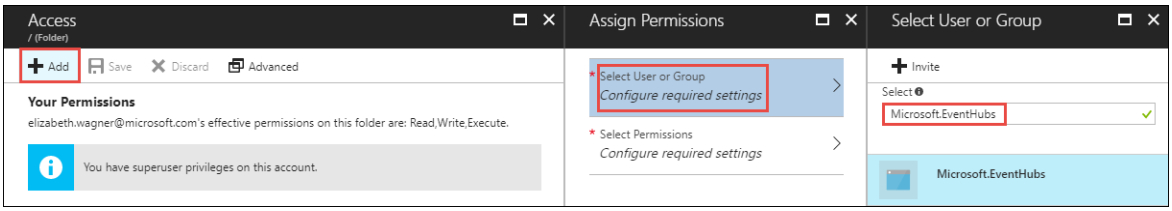
Click **OK**.

4. Assign permissions for the folder under the Data Lake Storage Gen1 account where you want to capture data.

a. Click **Data Explorer**, select the folder in the Data Lake Storage Gen1 account, and then click **Access**.

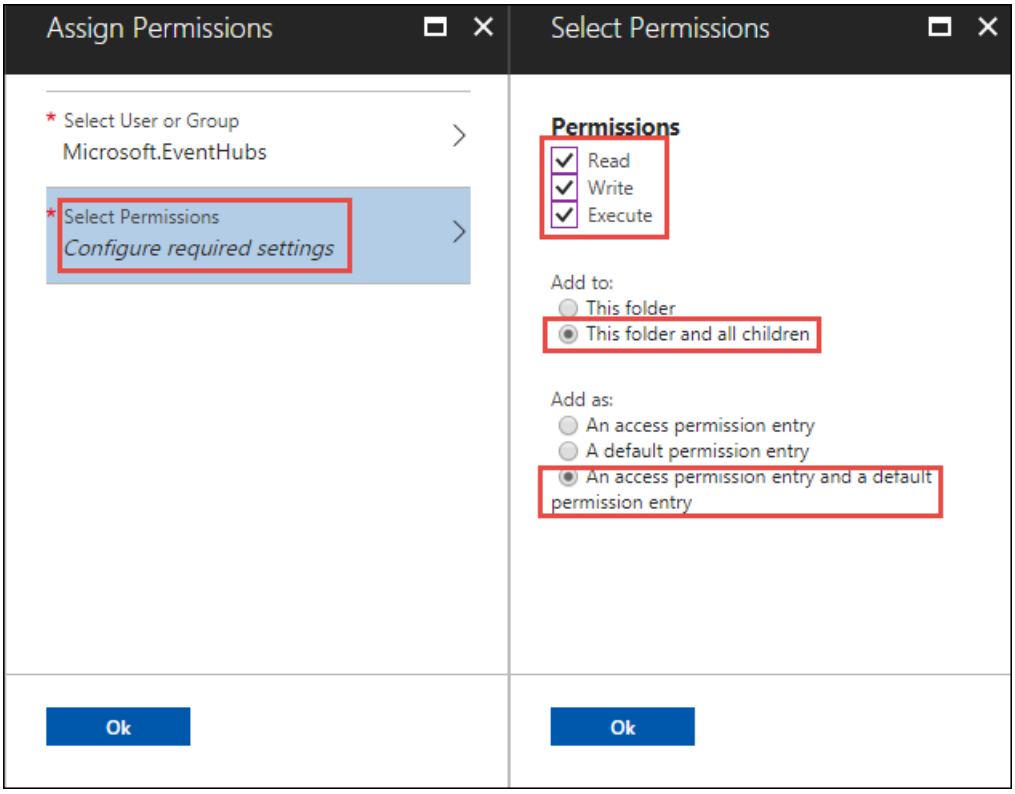


b. Under **Access**, click **Add**, click **Select User or Group**, and then search for `Microsoft.EventHubs`.



Click **Select**.

c. Under **Assign Permissions**, click **Select Permissions**. Set **Permissions** to **Read, Write,** and **Execute**. Set **Add to** to **This folder and all children**. Finally, set **Add as** to **An access permission entry and a default permission entry**.

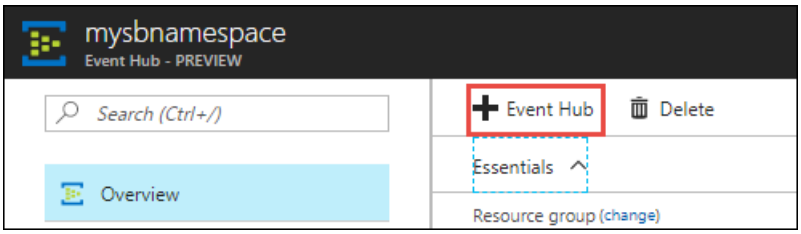


Click **OK**.

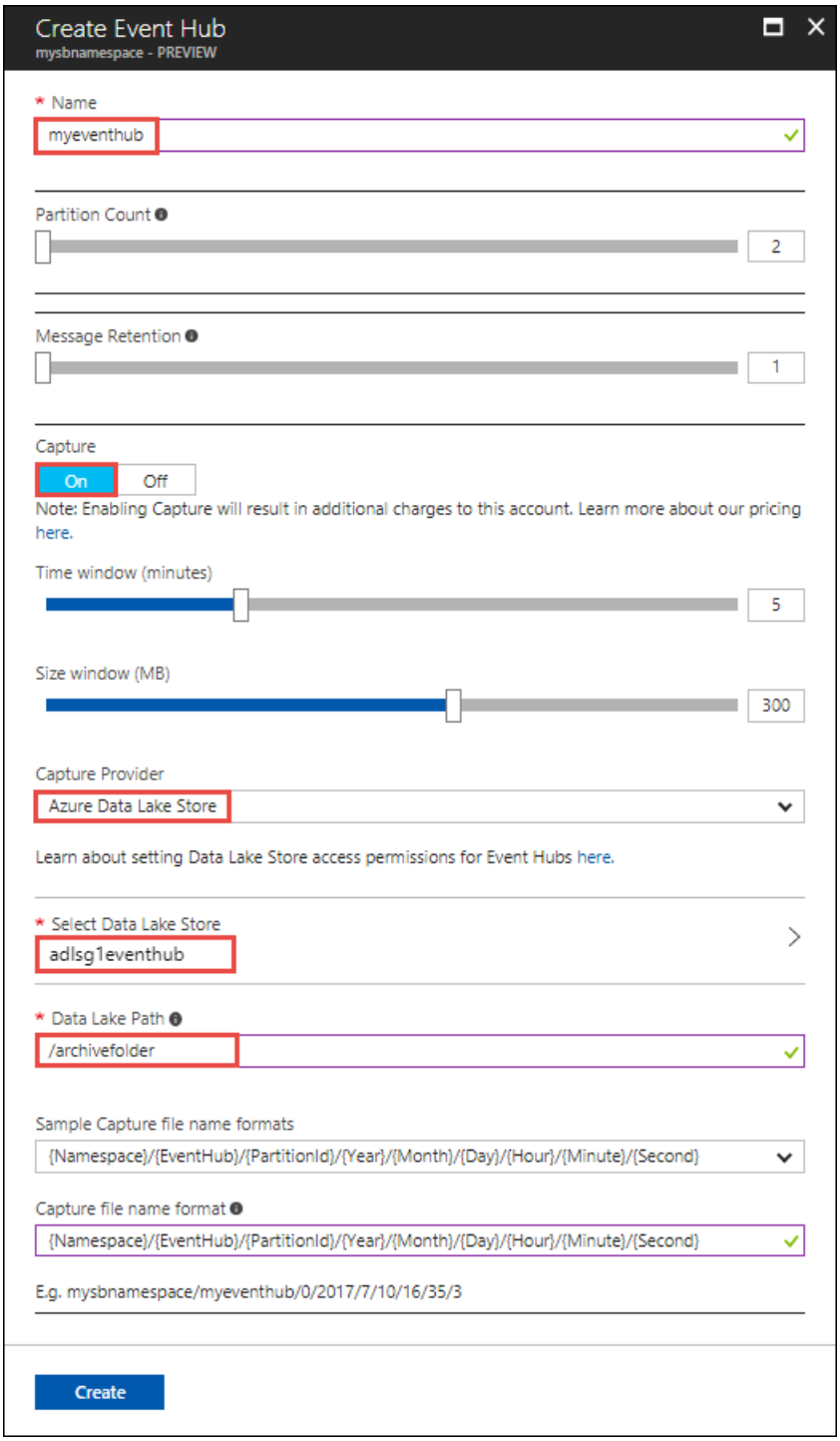
Configure Event Hubs to capture data to Data Lake Storage Gen1

In this section, you create an Event Hub within an Event Hubs namespace. You also configure the Event Hub to capture data to an Azure Data Lake Storage Gen1 account. This section assumes that you have already created an Event Hubs namespace.

1. From the **Overview** pane of the Event Hubs namespace, click **+ Event Hub**.



2. Provide the following values to configure Event Hubs to capture data to Data Lake Storage Gen1.



a. Provide a name for the Event Hub.

b. For this tutorial, set **Partition Count** and **Message Retention** to the default values.

c. Set **Capture** to **On**. Set the **Time Window** (how frequently to capture) and **Size Window** (data size to capture).

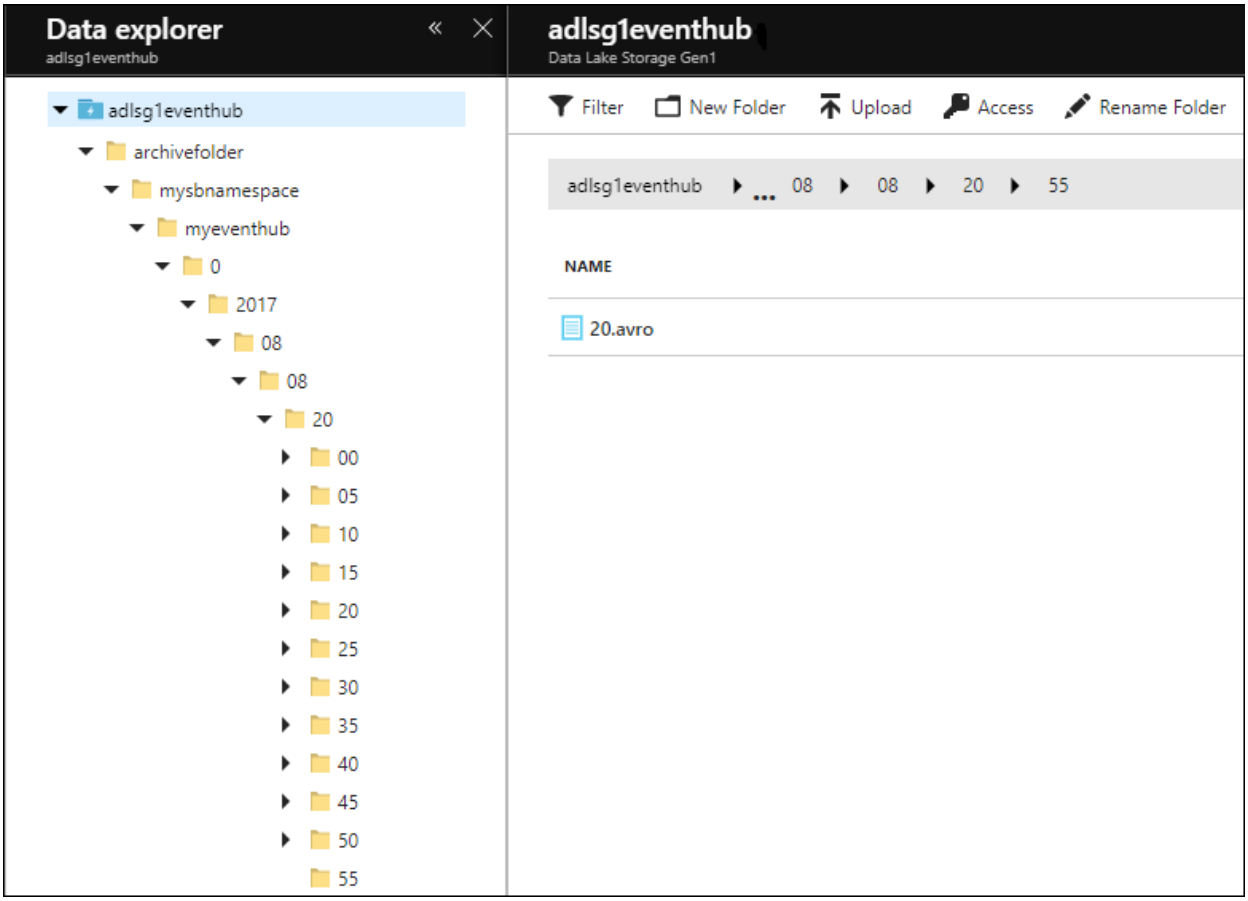
d. For **Capture Provider**, select **Azure Data Lake Store** and then select the Data Lake Storage Gen1 account you created earlier. For **Data Lake Path**, enter the name of the folder you created in the Data Lake Storage Gen1 account. You only need to provide the relative path to the folder.

e. Leave the **Sample capture file name formats** to the default value. This option governs the folder structure that is created under the capture folder.

f. Click **Create**.

Test the setup

You can now test the solution by sending data to the Azure Event Hub. Follow the instructions at Send events to Azure Event Hubs. Once you start sending the data, you see the data reflected in Data Lake Storage Gen1 using the folder structure you specified. For example, you see a folder structure, as shown in the following screenshot, in your Data Lake Storage Gen1 account.



NOTE

Even if you do not have messages coming into Event Hubs, Event Hubs writes empty files with just the headers into the Data Lake Storage Gen1 account. The files are written at the same time interval that you provided while creating the Event Hubs.

Analyze data in Data Lake Storage Gen1

Once the data is in Data Lake Storage Gen1, you can run analytical jobs to process and crunch the data. See USQL Avro Example on how to do this using Azure Data Lake Analytics.

See also

- Secure data in Data Lake Storage Gen1

- Copy data from Azure Storage Blobs to Data Lake Storage Gen1

Load data into Azure Data Lake Storage Gen1 by using Azure Data Factory

10/3/2022 • 4 minutes to read • Edit Online

APPLIES TO: Azure Data Factory Azure Synapse Analytics

Azure Data Lake Storage Gen1 (previously known as Azure Data Lake Store) is an enterprise-wide hyper-scale repository for big data analytic workloads. Data Lake Storage Gen1 lets you capture data of any size, type, and ingestion speed. The data is captured in a single place for operational and exploratory analytics.

Azure Data Factory is a fully managed cloud-based data integration service. You can use the service to populate the lake with data from your existing system and save time when building your analytics solutions.

Azure Data Factory offers the following benefits for loading data into Data Lake Storage Gen1:

- **Easy to set up**: An intuitive 5-step wizard with no scripting required.
- **Rich data store support**: Built-in support for a rich set of on-premises and cloud-based data stores. For a detailed list, see the table of Supported data stores.
- **Secure and compliant**: Data is transferred over HTTPS or ExpressRoute. The global service presence ensures that your data never leaves the geographical boundary.
- **High performance**: Up to 1-GB/s data loading speed into Data Lake Storage Gen1. For details, see Copy activity performance.

This article shows you how to use the Data Factory Copy Data tool to *load data from Amazon S3 into Data Lake Storage Gen1*. You can follow similar steps to copy data from other types of data stores.

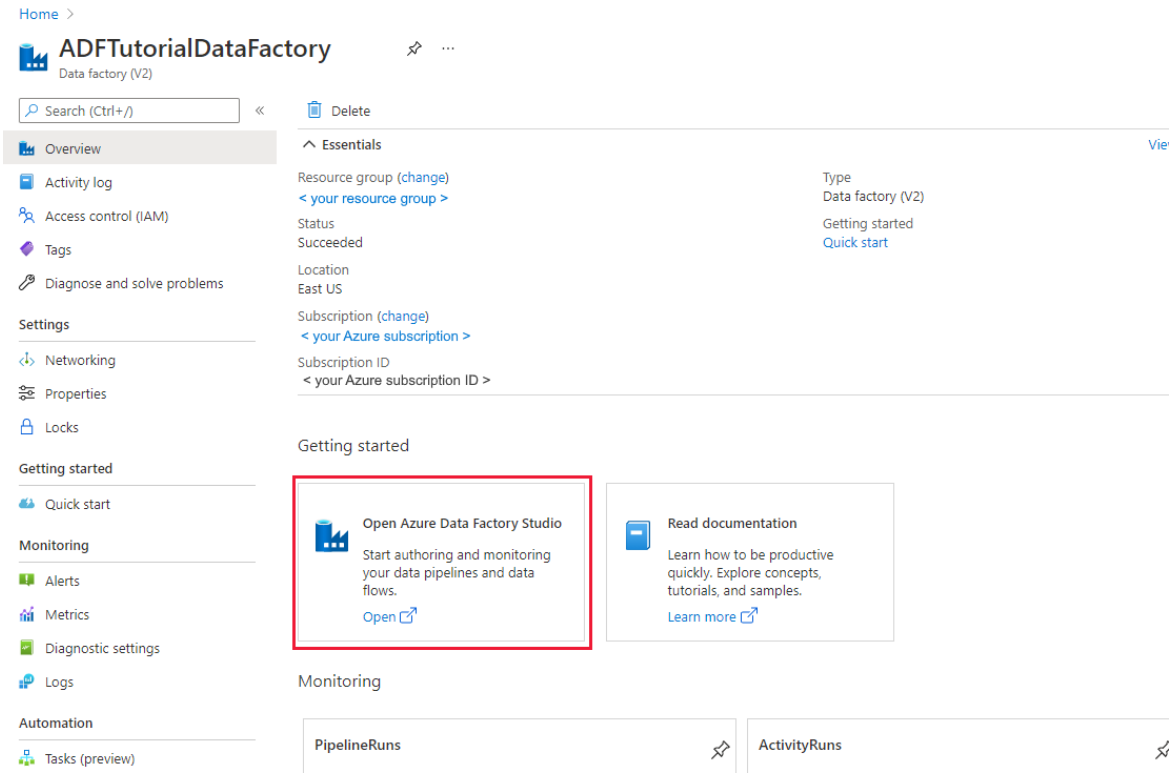
> **NOTE**
> For more information, see Copy data to or from Data Lake Storage Gen1 by using Azure Data Factory.

Prerequisites

- Azure subscription: If you don't have an Azure subscription, create a free account before you begin.
- Data Lake Storage Gen1 account: If you don't have a Data Lake Storage Gen1 account, see the instructions in Create a Data Lake Storage Gen1 account.
- Amazon S3: This article shows how to copy data from Amazon S3. You can use other data stores by following similar steps.

Create a data factory

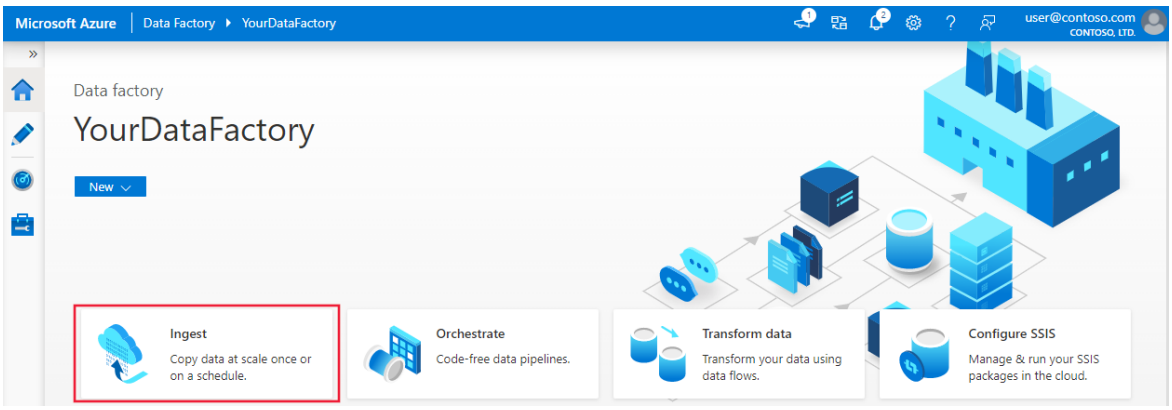
1. If you have not created your data factory yet, follow the steps in Quickstart: Create a data factory by using the Azure portal and Azure Data Factory Studio to create one. After creating it, browse to the data factory in the Azure portal.



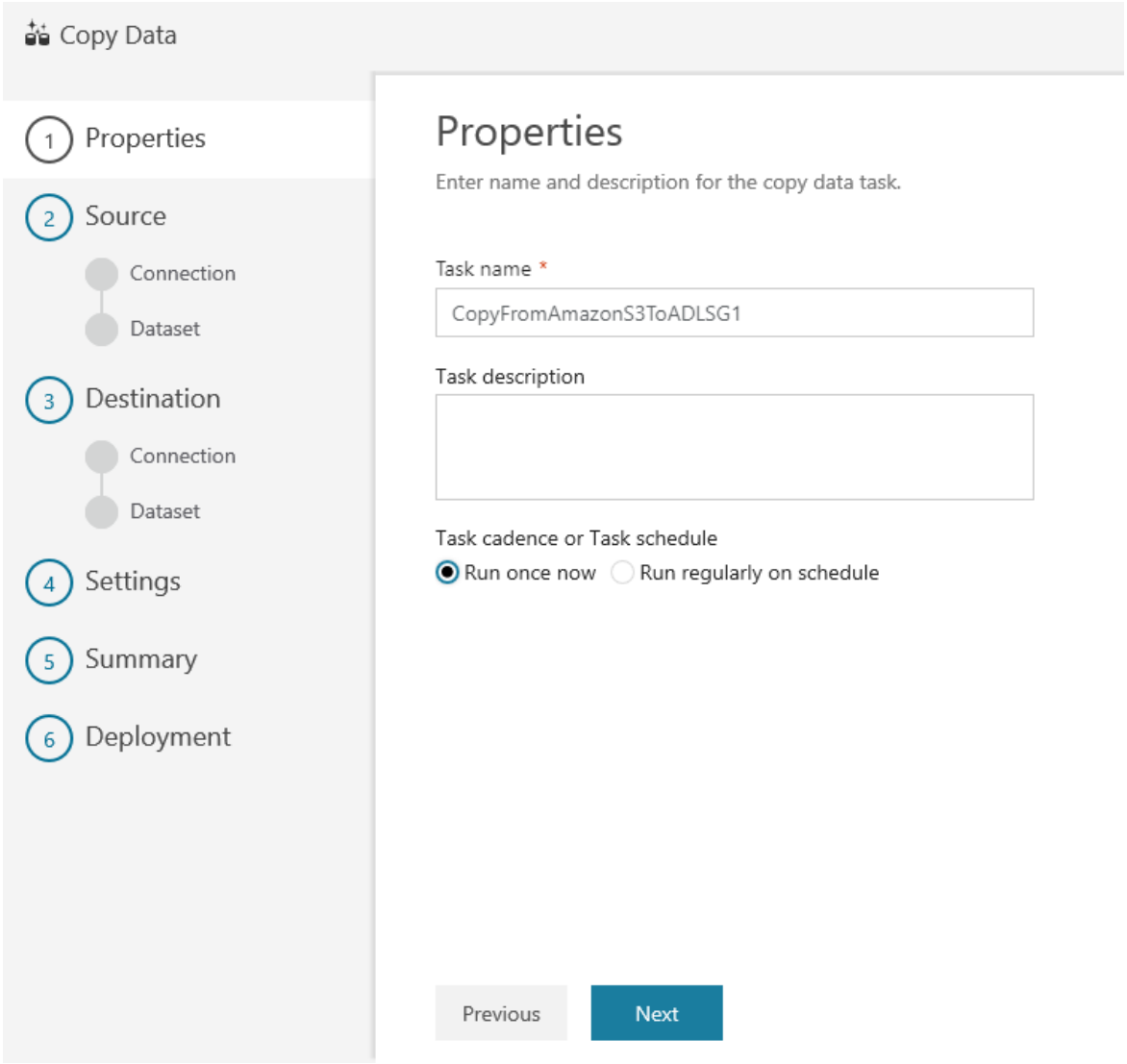
2. Select **Open** on the **Open Azure Data Factory Studio** tile to launch the Data Integration application in a separate tab.

Load data into Data Lake Storage Gen1

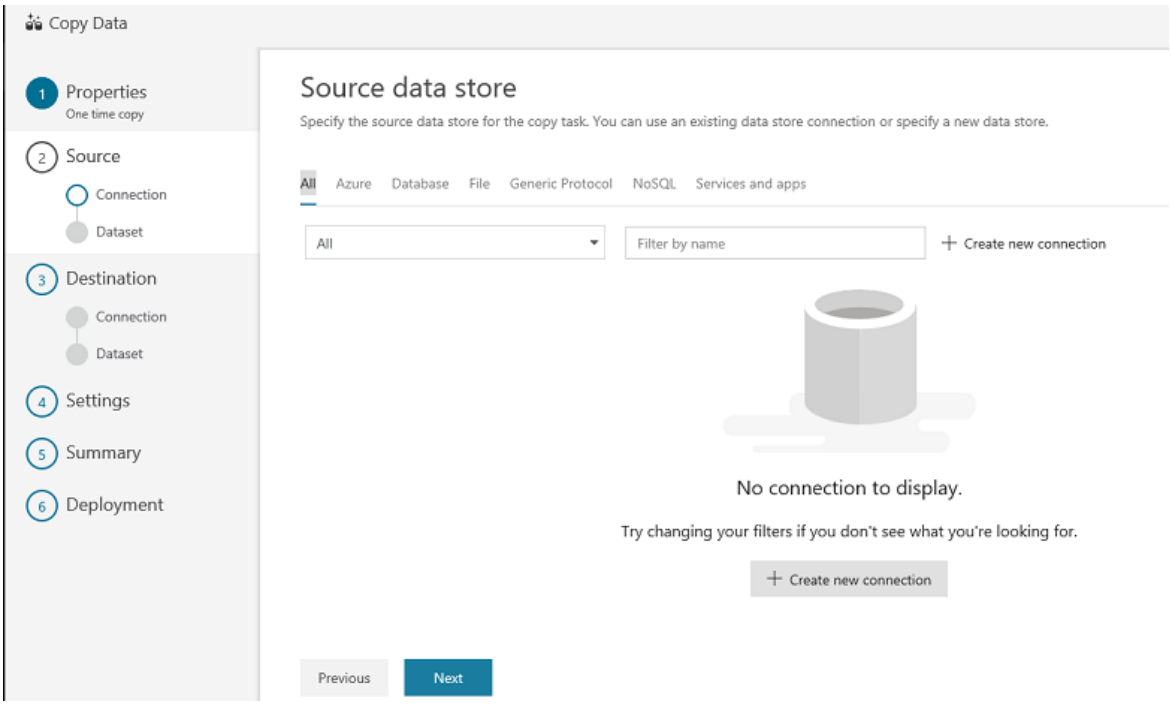
1. In the home page, select the **Ingest** tile to launch the Copy Data tool:



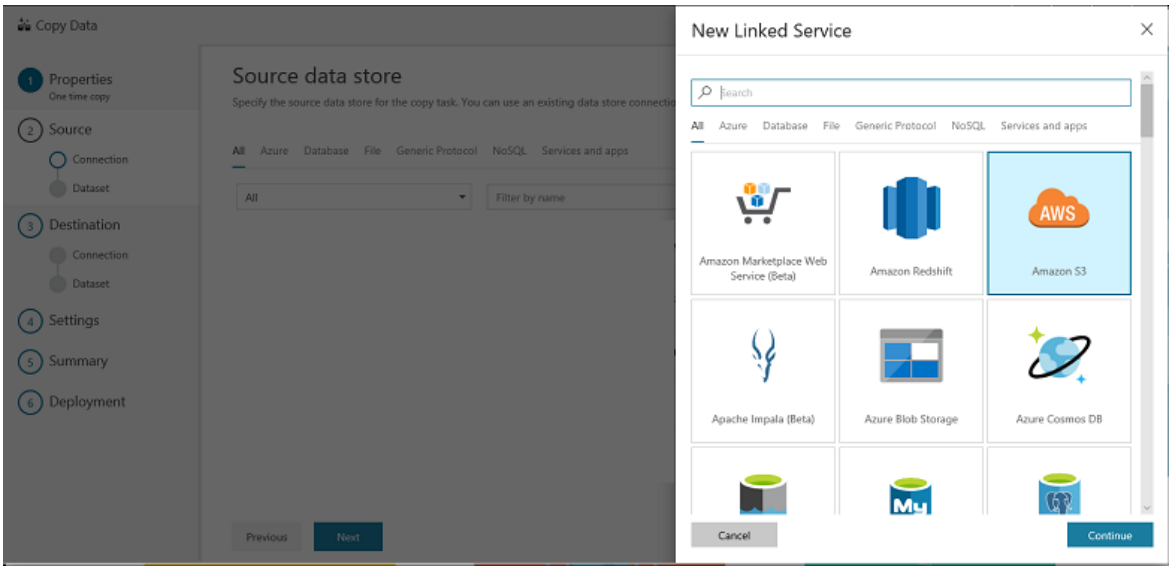
2. In the **Properties** page, specify **CopyFromAmazonS3ToADLS** for the **Task name** field, and select **Next**:



3. In the **Source data store** page, select **+ Create new connection**:



Select **Amazon S3**, and select **Continue**

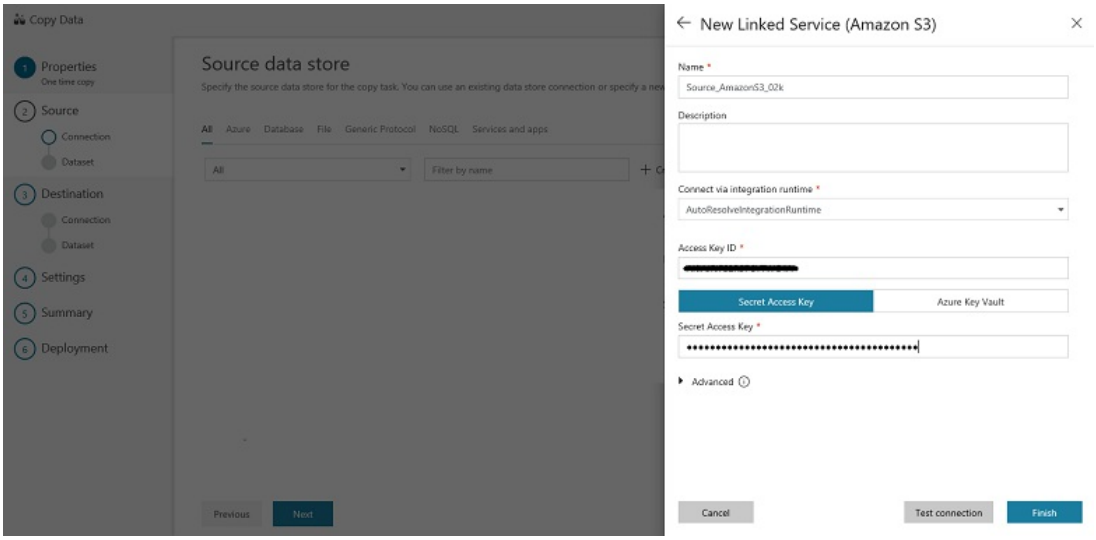


4. In the **Specify Amazon S3 connection** page, do the following steps:

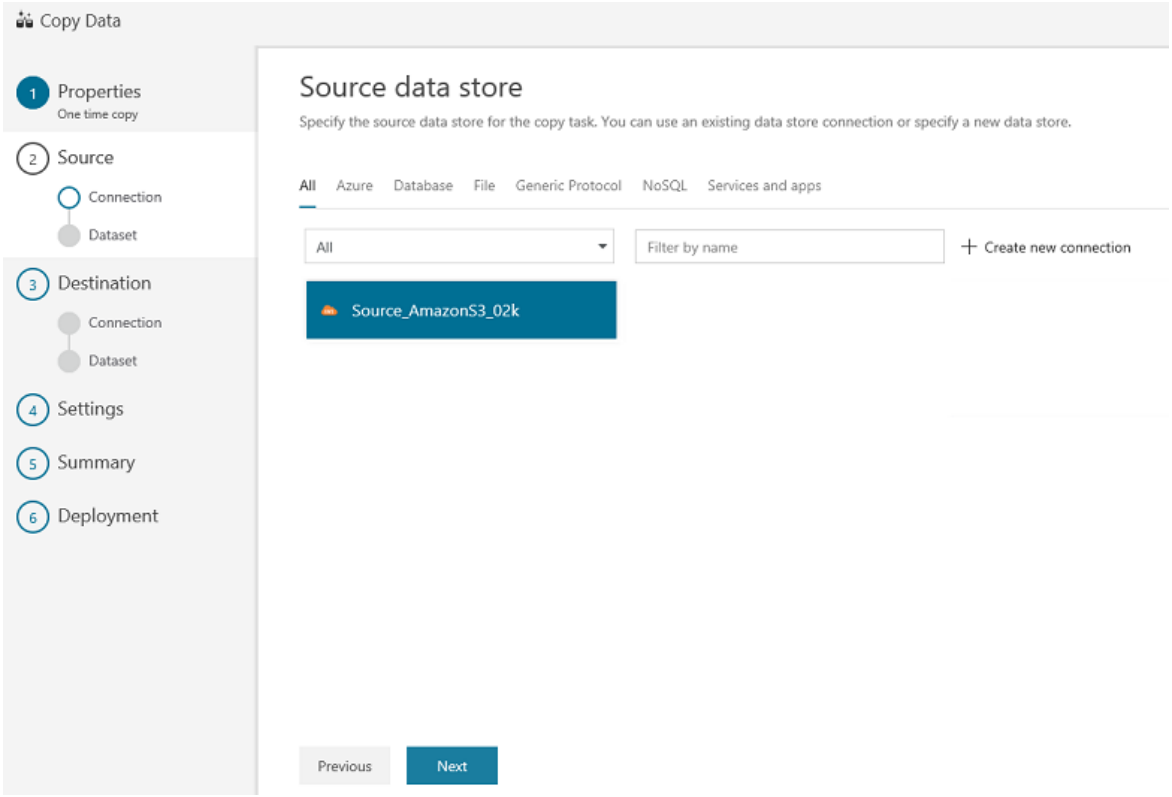
   a. Specify the **Access Key ID** value.

   b. Specify the **Secret Access Key** value.

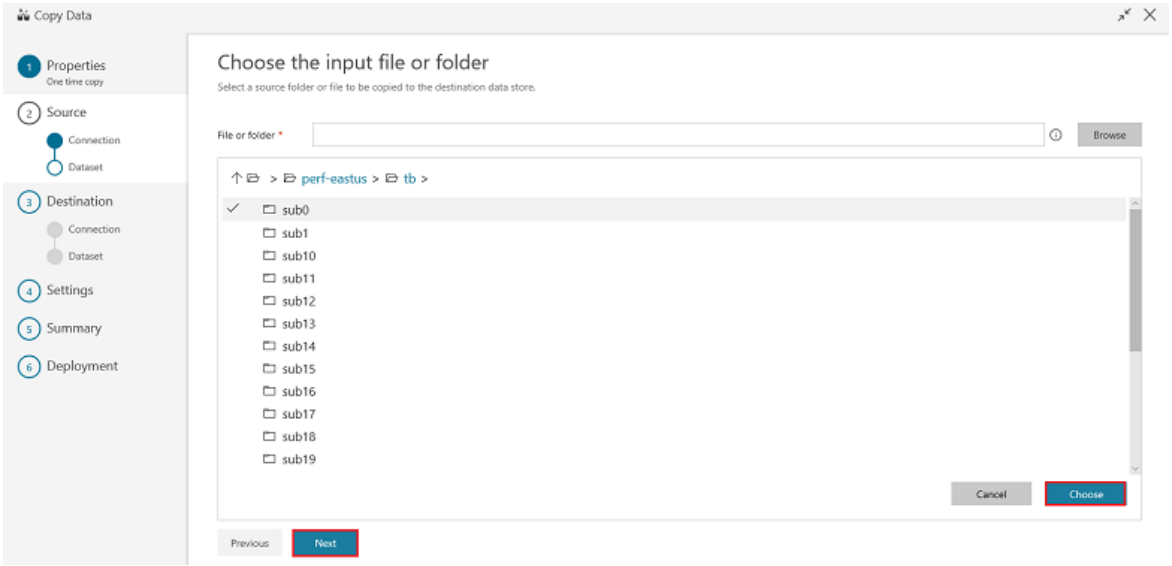
   c. Select **Finish**.



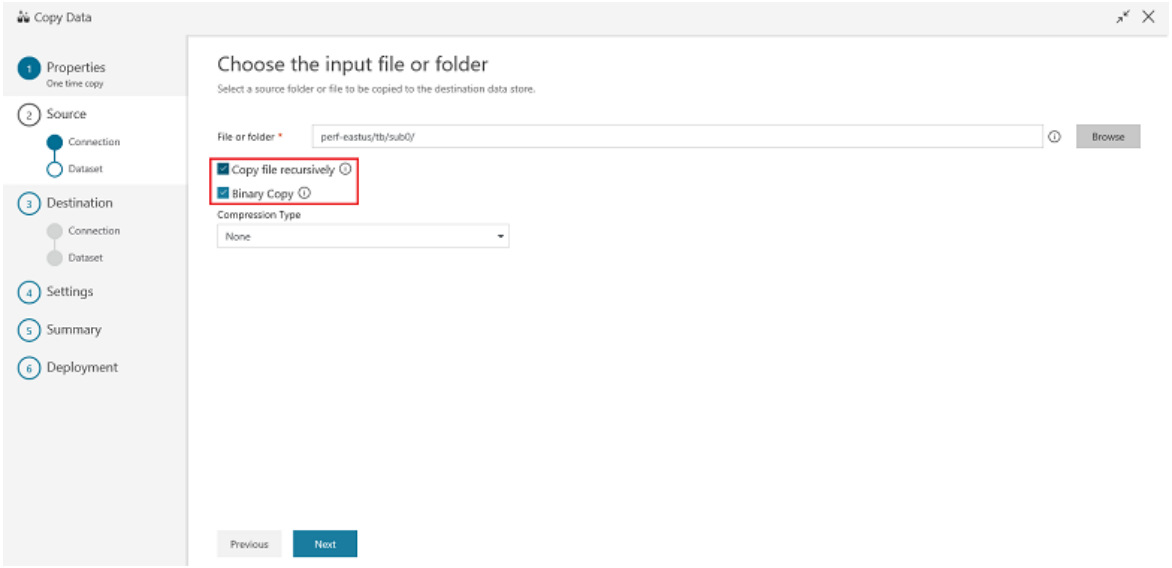
   d. You will see a new connection. Select **Next**.



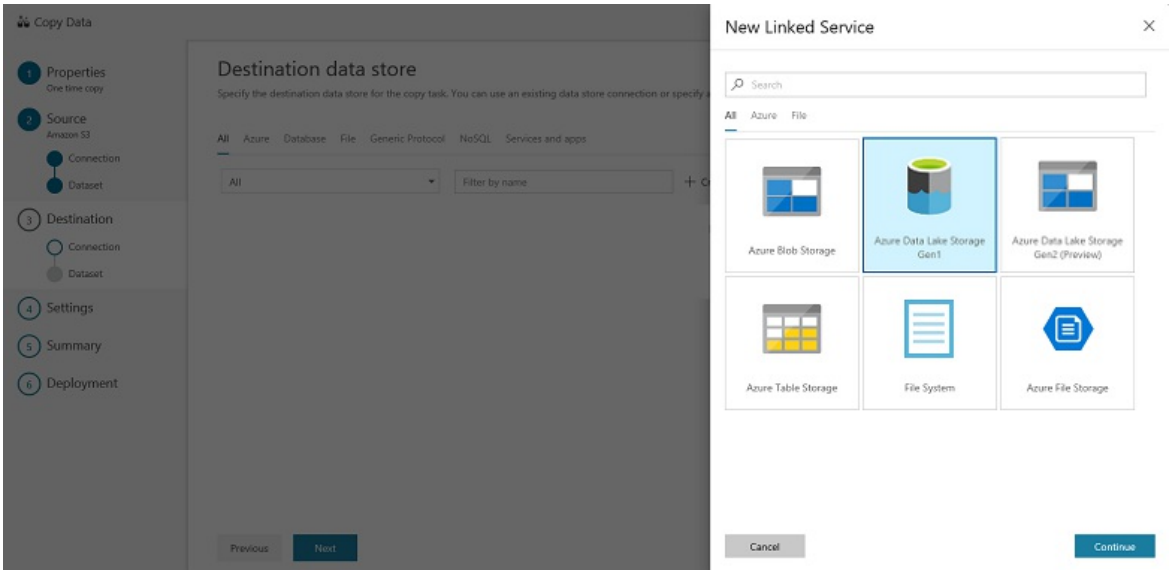
5. In the **Choose the input file or folder** page, browse to the folder and file that you want to copy over. Select the folder/file, select **Choose**, and then select **Next**:



6. Choose the copy behavior by selecting the **Copy files recursively** and **Binary copy** (copy files as-is) options. Select **Next**:



7. In the **Destination data store** page, select **+ Create new connection**, and then select **Azure Data Lake Storage Gen1**, and select **Continue**:



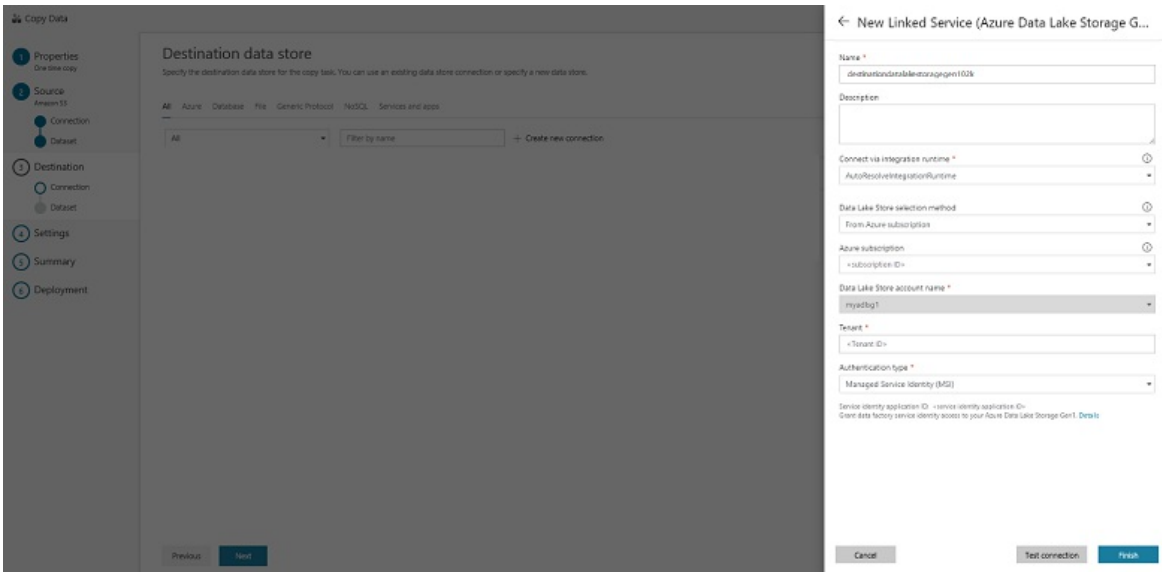
8. In the **New Linked Service (Azure Data Lake Storage Gen1)** page, do the following steps:

   a. Select your Data Lake Storage Gen1 account for the **Data Lake Store account name**.

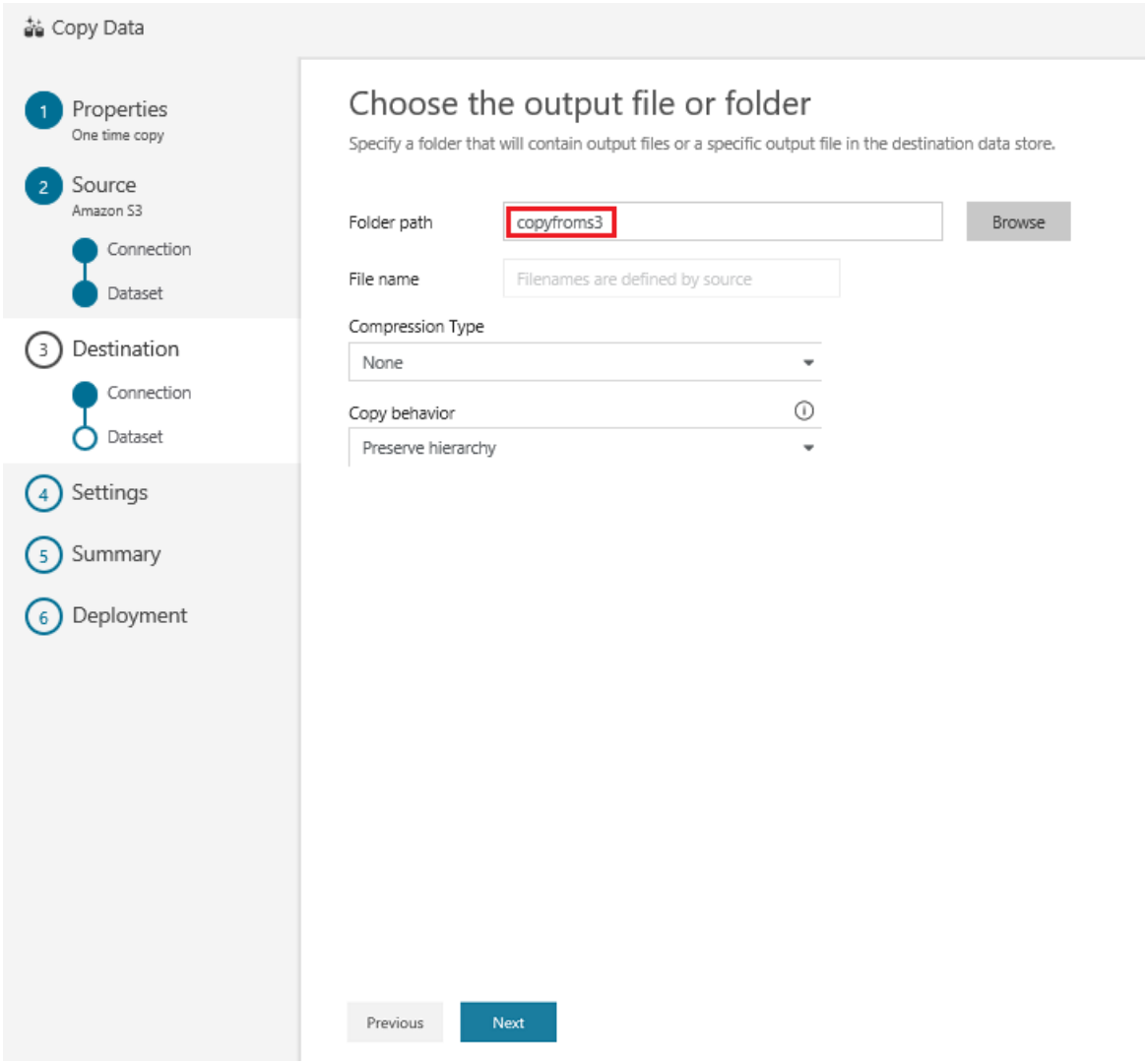
   b. Specify the **Tenant**, and select Finish.

   c. Select **Next**.

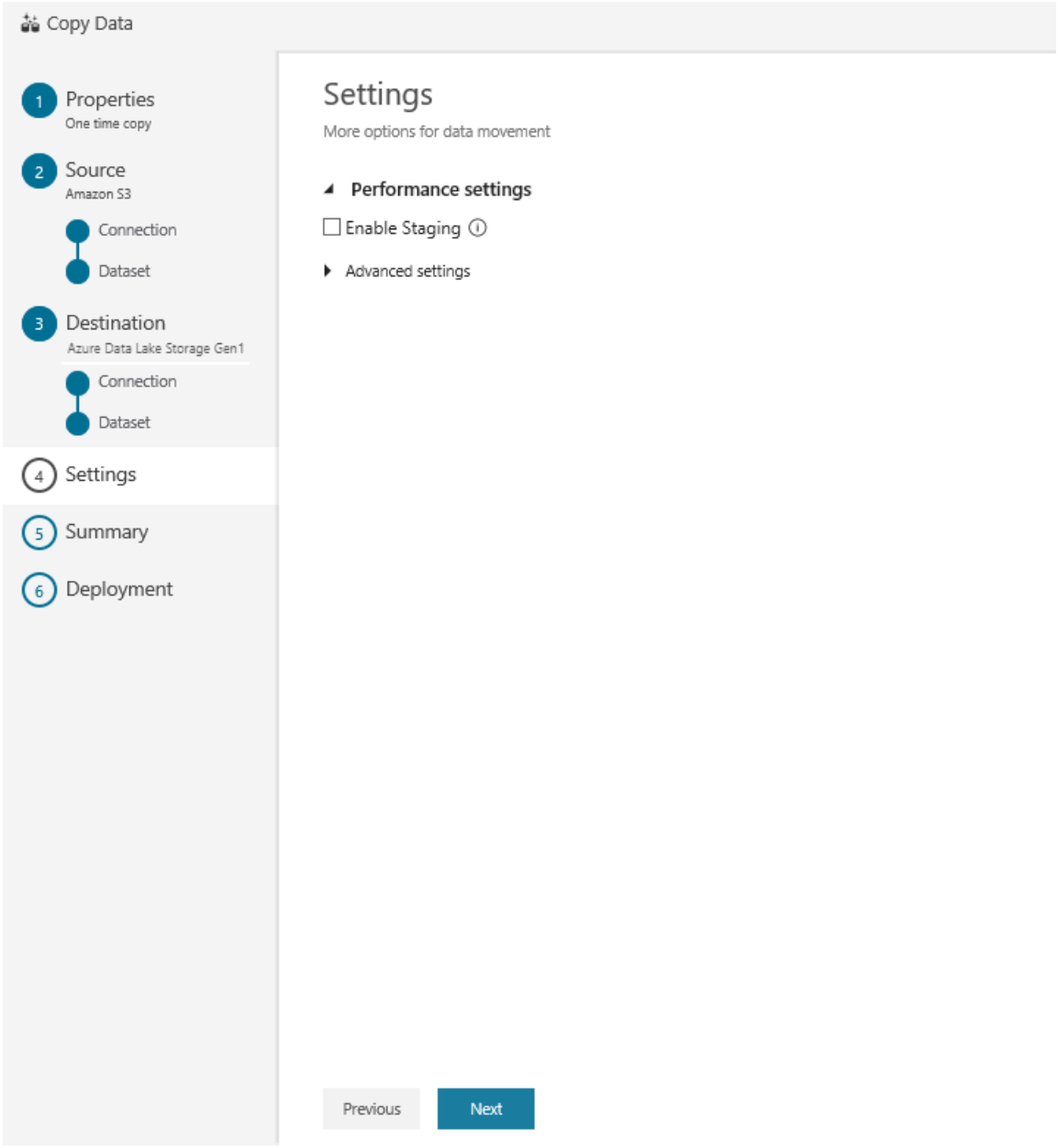
> **IMPORTANT**
>
> In this walkthrough, you use a managed identity for Azure resources to authenticate your Data Lake Storage Gen1 account. Be sure to grant the MSI the proper permissions in Data Lake Storage Gen1 by following these instructions.



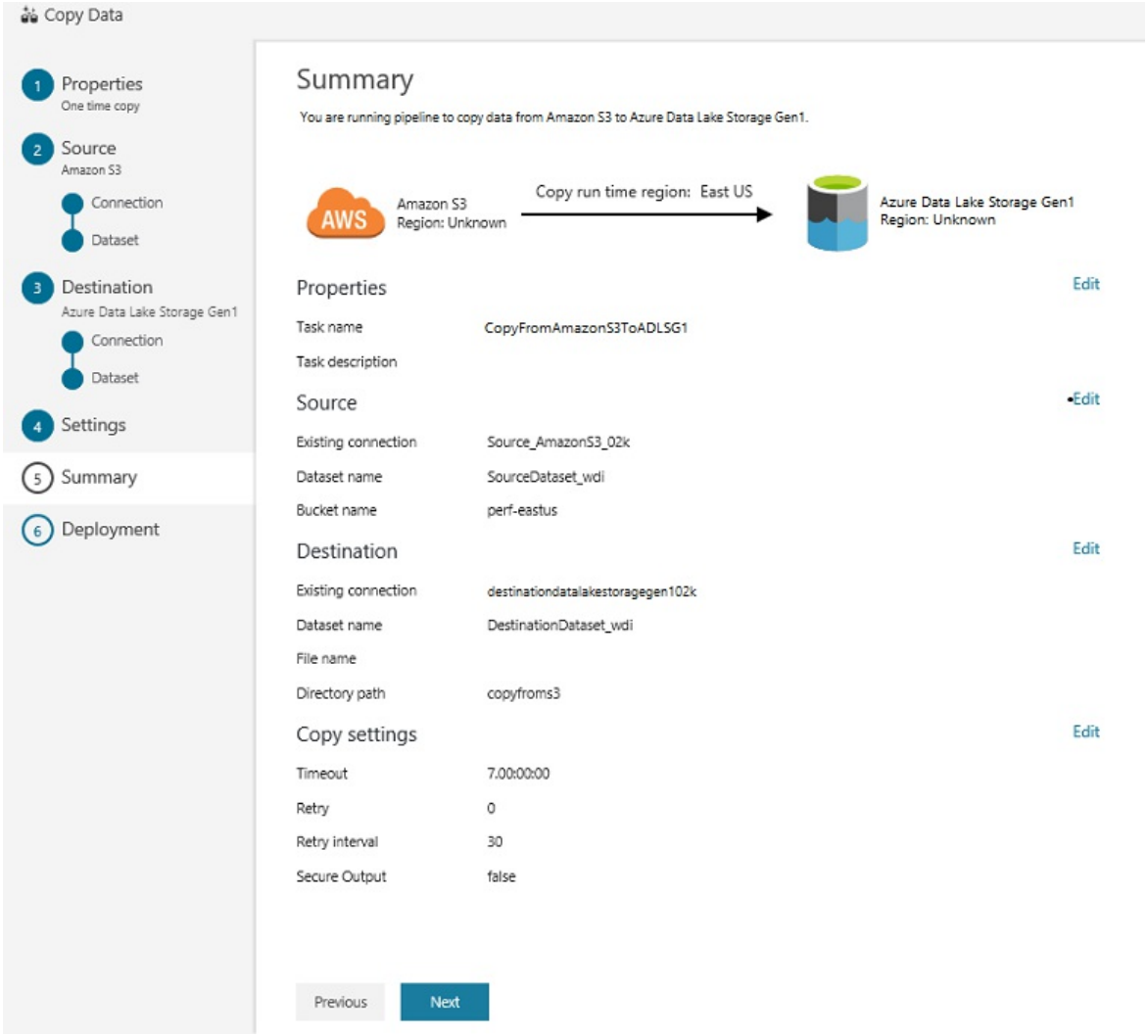
9. In the **Choose the output file or folder** page, enter **copyfroms3** as the output folder name, and select **Next**:



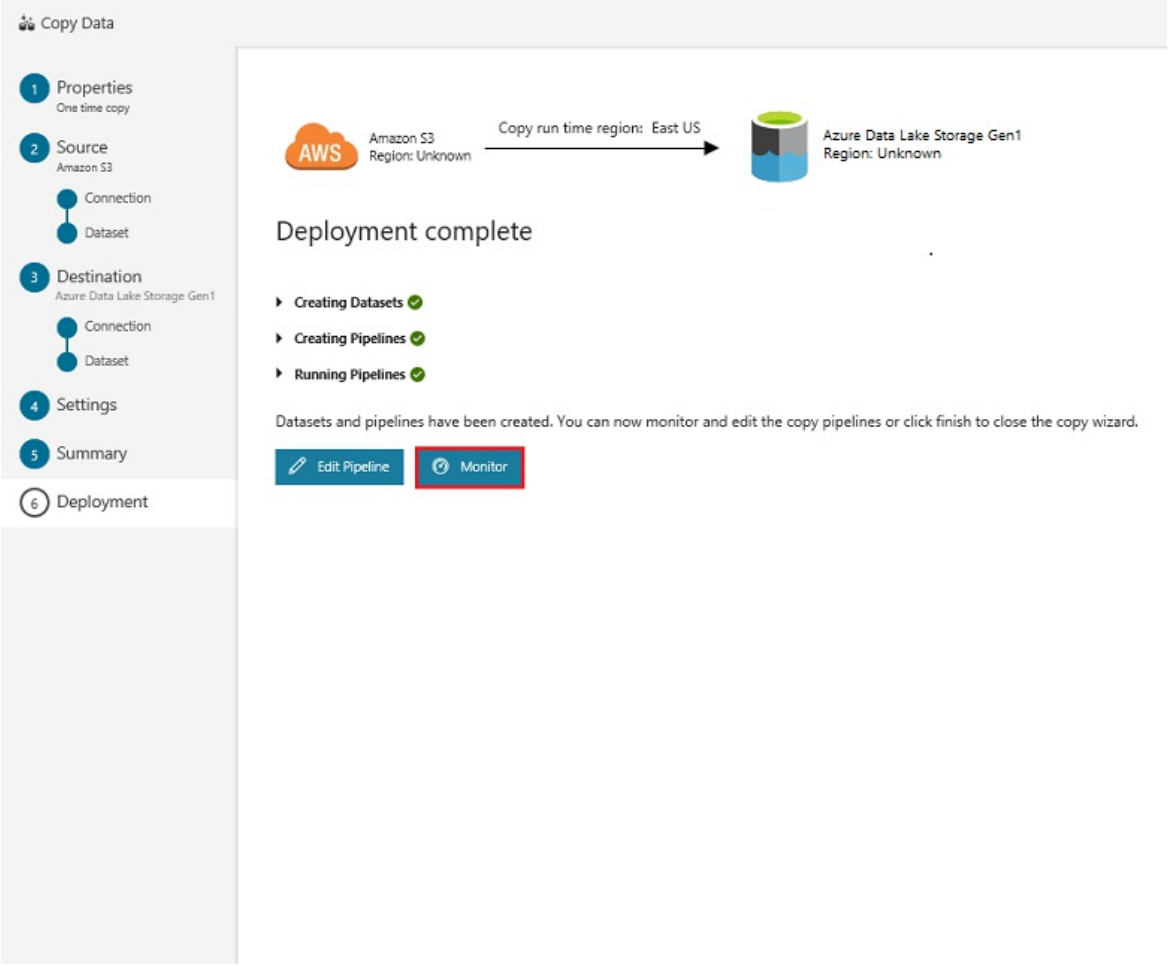
10. In the **Settings** page, select **Next**:



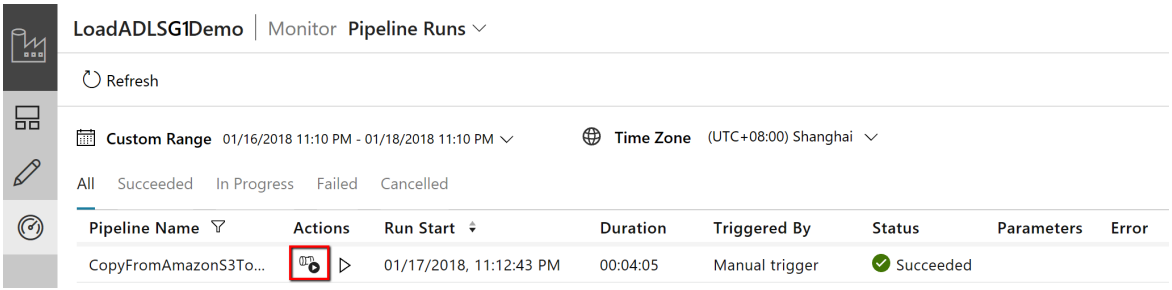
11. In the **Summary** page, review the settings, and select **Next**:



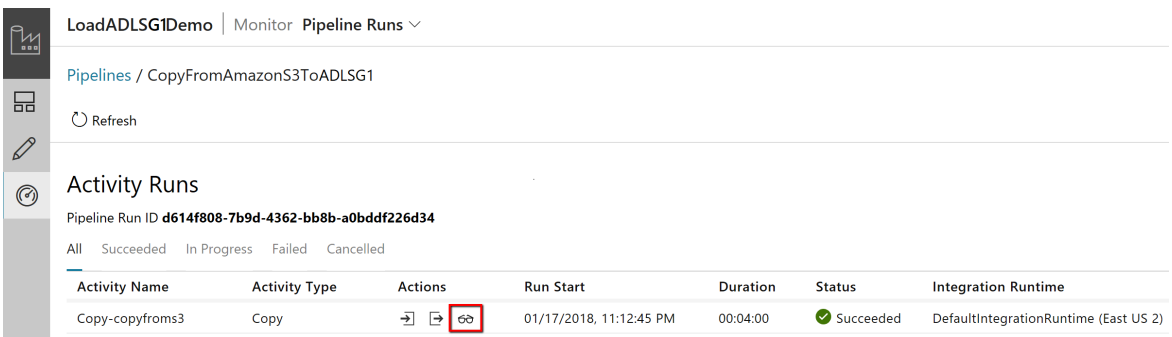
12. In the **Deployment page**, select **Monitor** to monitor the pipeline (task):



13. Notice that the **Monitor** tab on the left is automatically selected. The **Actions** column includes links to view activity run details and to rerun the pipeline:



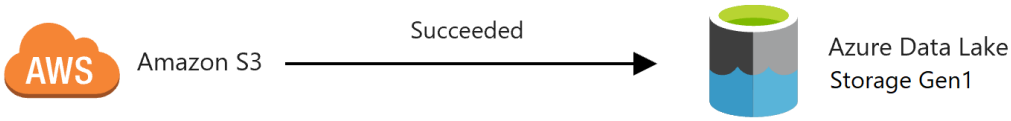
14. To view activity runs that are associated with the pipeline run, select the **View Activity Runs** link in the **Actions** column. There's only one activity (copy activity) in the pipeline, so you see only one entry. To switch back to the pipeline runs view, select the **Pipelines** link at the top. Select **Refresh** to refresh the list.



15. To monitor the execution details for each copy activity, select the **Details** link under **Actions** in the activity monitoring view. You can monitor details like the volume of data copied from the source to the sink, data throughput, execution steps with corresponding duration, and used configurations:

Details

Learn more on copy performance details from here.



Data read: 107.281 GB
Files read: 10

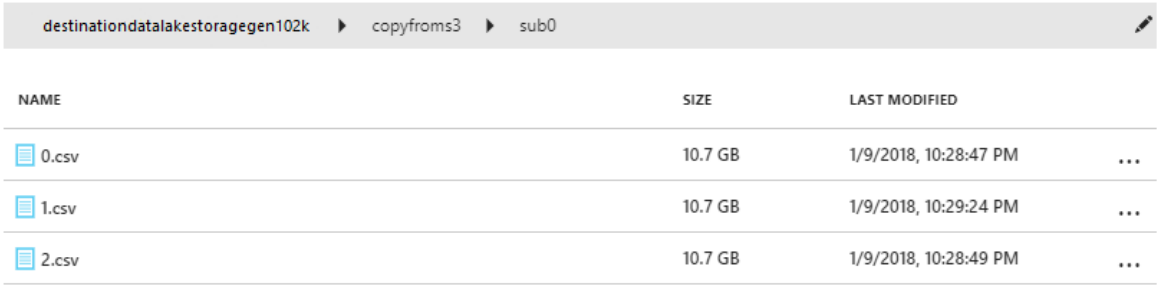
Data written: 107.281 GB
Files written: 10
Throughput: 467.707 MB/s

Execution details:

Amazon S3 → Azure Data Lake Storage Gen1 Queue 00:00:02 | Transfer 00:03:39

Start time	01/17/2018, 11:13:00 PM
Duration	00:03:41
Used DMUs	32
Used parallel copies	8

16. Verify that the data is copied into your Data Lake Storage Gen1 account:



Next steps

Advance to the following article to learn about Data Lake Storage Gen1 support:

Azure Data Lake Storage Gen1 connector

Stream data from Azure Storage Blob into Azure Data Lake Storage Gen1 using Azure Stream Analytics

10/3/2022 • 3 minutes to read • Edit Online

In this article, you learn how to use Azure Data Lake Storage Gen1 as an output for an Azure Stream Analytics job. This article demonstrates a simple scenario that reads data from an Azure Storage blob (input) and writes the data to Data Lake Storage Gen1 (output).

Prerequisites

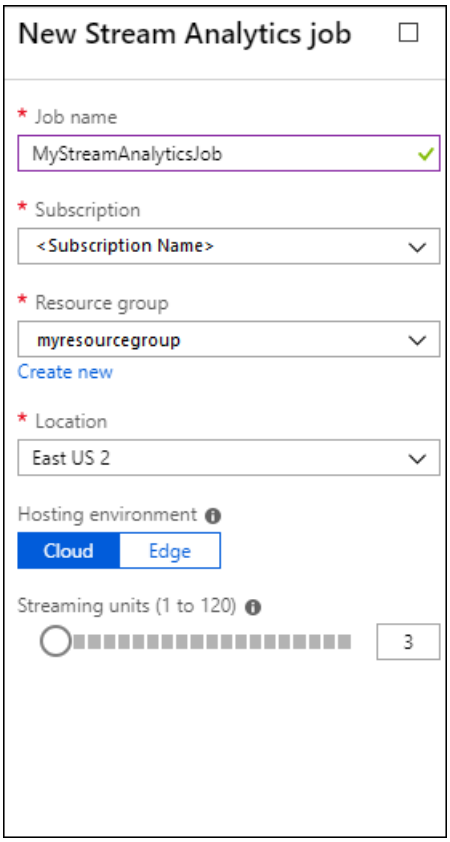
Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Azure Storage account**. You will use a blob container from this account to input data for a Stream Analytics job. For this tutorial, assume you have a storage account called **storageforasa** and a container within the account called **storageforasacontainer**. Once you have created the container, upload a sample data file to it.
- **A Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal. Let's assume you have a Data Lake Storage Gen1 account called **myadlsg1**.

Create a Stream Analytics Job

You start by creating a Stream Analytics job that includes an input source and an output destination. For this tutorial, the source is an Azure blob container and the destination is Data Lake Storage Gen1.

1. Sign on to the Azure portal.
2. From the left pane, click **Stream Analytics jobs**, and then click **Add**.

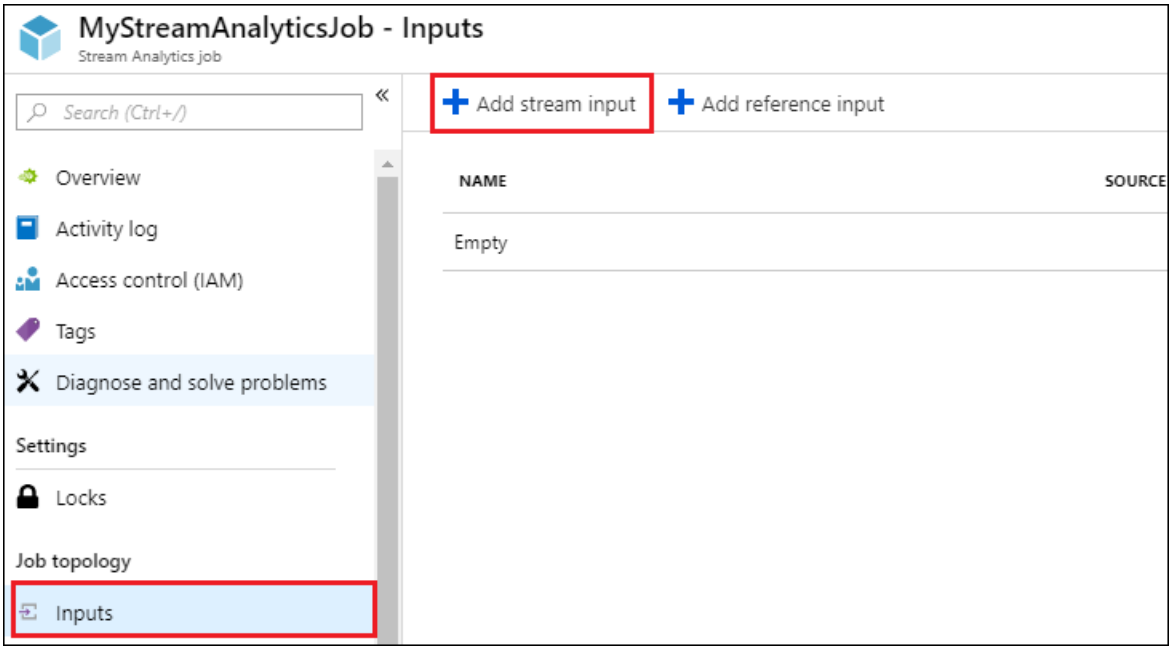


NOTE

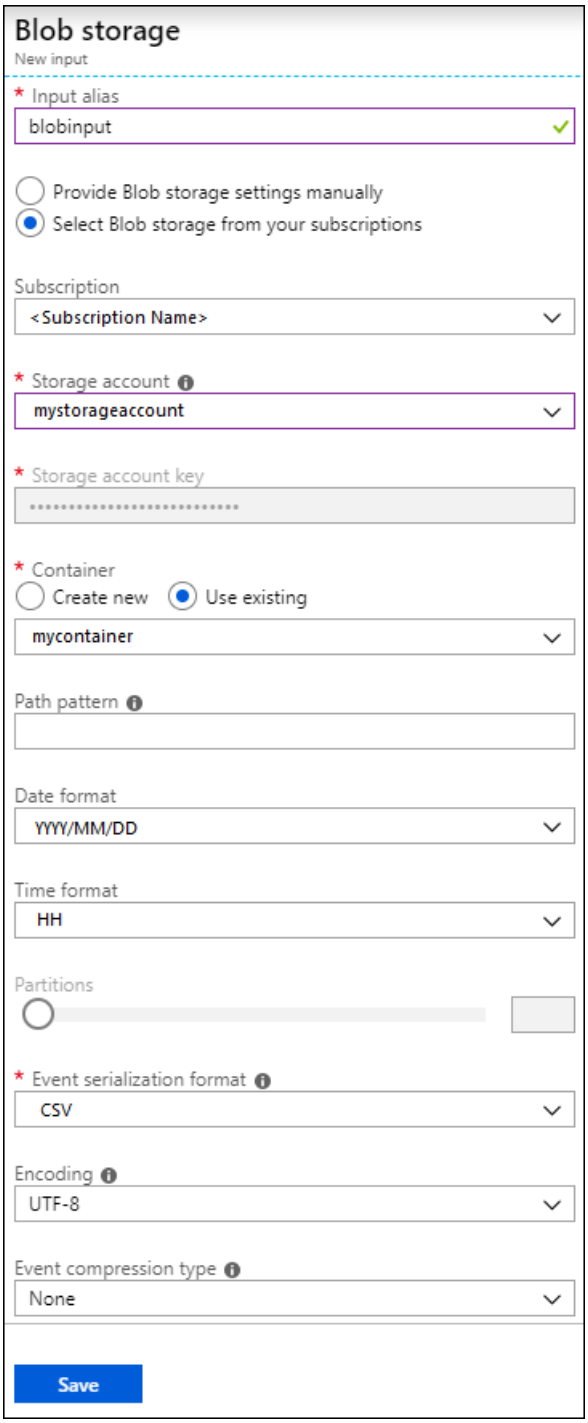
Make sure you create job in the same region as the storage account or you will incur additional cost of moving data between regions.

Create a Blob input for the job

1. Open the page for the Stream Analytics job, from the left pane click the **Inputs** tab, and then click **Add**.



2. On the **New input** blade, provide the following values.

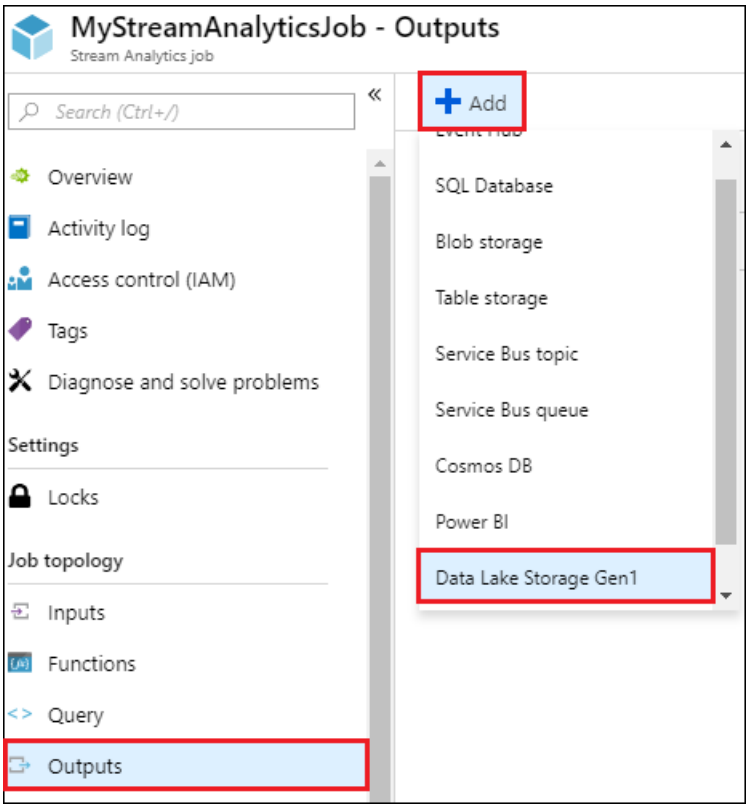


- For **Input alias**, enter a unique name for the job input.
- For **Source type**, select **Data stream**.
- For **Source**, select **Blob storage**.
- For **Subscription**, select **Use blob storage from current subscription**.
- For **Storage account**, select the storage account that you created as part of the prerequisites.
- For **Container**, select the container that you created in the selected storage account.
- For **Event serialization format**, select **CSV**.
- For **Delimiter**, select **tab**.
- For **Encoding**, select **UTF-8**.

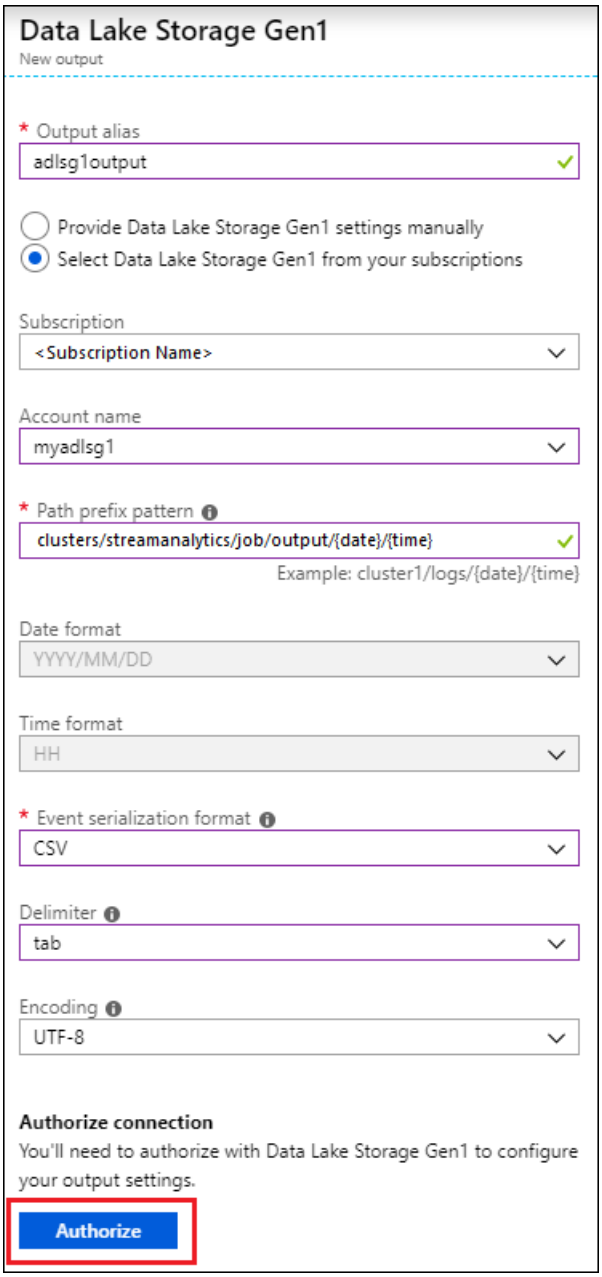
  Click **Create**. The portal now adds the input and tests the connection to it.

Create a Data Lake Storage Gen1 output for the job

1. Open the page for the Stream Analytics job, click the **Outputs** tab, click **Add**, and select **Data Lake Storage Gen1**.

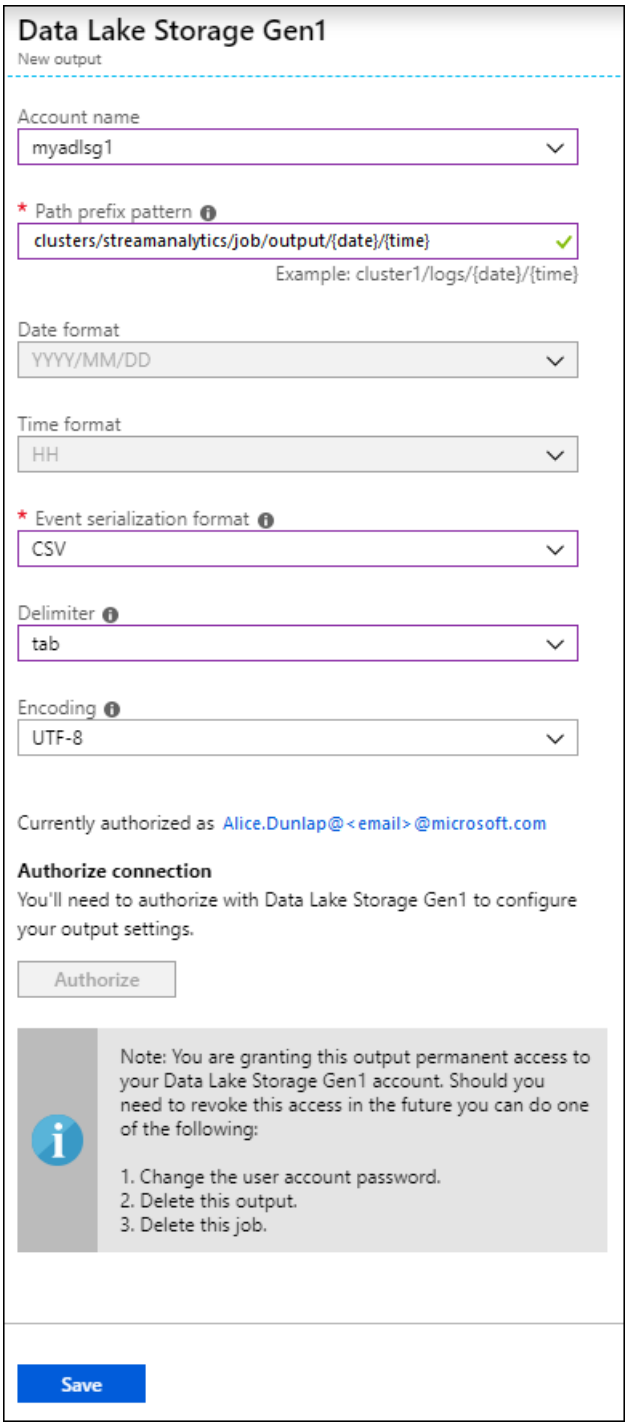


2. On the **New output** blade, provide the following values.



- For **Output alias**, enter a unique name for the job output. This is a friendly name used in queries to direct the query output to this Data Lake Storage Gen1 account.
- You will be prompted to authorize access to the Data Lake Storage Gen1 account. Click **Authorize**.

3. On the **New output** blade, continue to provide the following values.

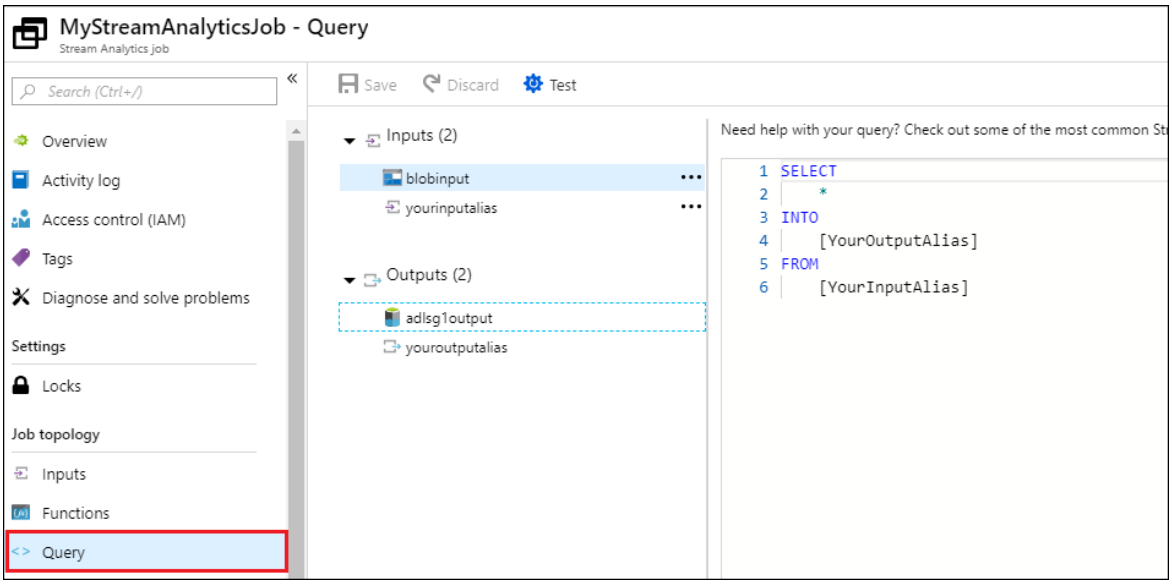


- For **Account name**, select the Data Lake Storage Gen1 account you already created where you want the job output to be sent to.
- For **Path prefix pattern**, enter a file path used to write your files within the specified Data Lake Storage Gen1 account.
- For **Date format**, if you used a date token in the prefix path, you can select the date format in which your files are organized.
- For **Time format**, if you used a time token in the prefix path, specify the time format in which your files are organized.
- For **Event serialization format**, select **CSV**.
- For **Delimiter**, select **tab**.
- For **Encoding**, select **UTF-8**.

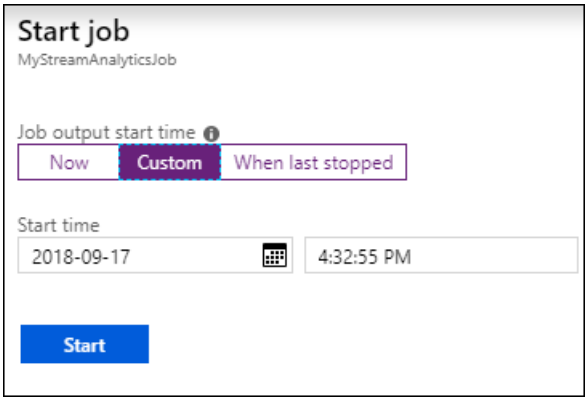
  Click **Create**. The portal now adds the output and tests the connection to it.

Run the Stream Analytics job

1. To run a Stream Analytics job, you must run a query from the **Query** tab. For this tutorial, you can run the sample query by replacing the placeholders with the job input and output aliases, as shown in the screen capture below.



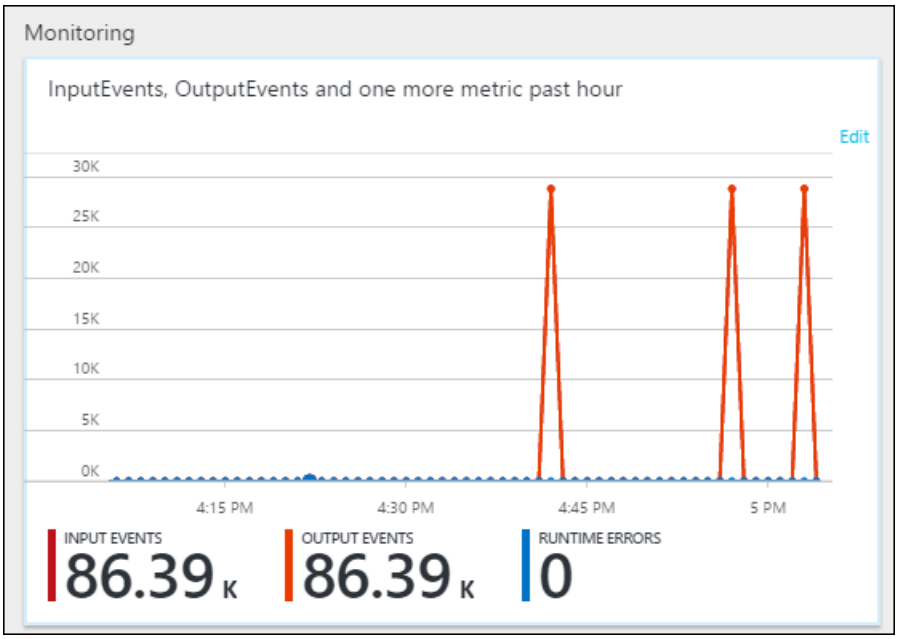
2. Click **Save** from the top of the screen, and then from the **Overview** tab, click **Start**. From the dialog box, select **Custom Time**, and then set the current date and time.



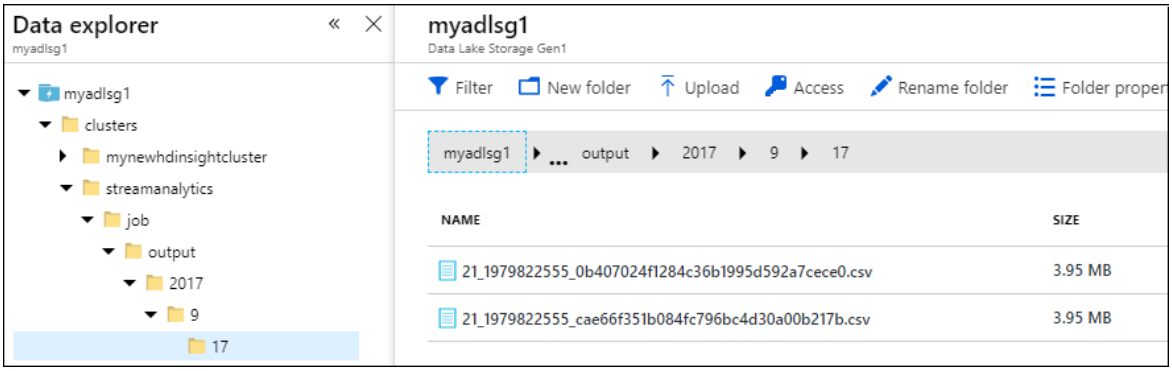
   Click **Start** to start the job. It can take up to a couple minutes to start the job.

3. To trigger the job to pick the data from the blob, copy a sample data file to the blob container. You can get a sample data file from the Azure Data Lake Git Repository. For this tutorial, let's copy the file **vehicle1_09142014.csv**. You can use various clients, such as Azure Storage Explorer, to upload data to a blob container.

4. From the **Overview** tab, under **Monitoring**, see how the data was processed.



5. Finally, you can verify that the job output data is available in the Data Lake Storage Gen1 account.



In the Data Explorer pane, notice that the output is written to a folder path as specified in the Data Lake Storage Gen1 output settings (`streamanalytics/job/output/{date}/{time}`).

See also

- Create an HDInsight cluster to use Data Lake Storage Gen1

Analyze data in Azure Data Lake Storage Gen1 by using Power BI

10/3/2022 • 3 minutes to read • Edit Online

In this article, you learn how to use Power BI Desktop to analyze and visualize data stored in Azure Data Lake Storage Gen1.

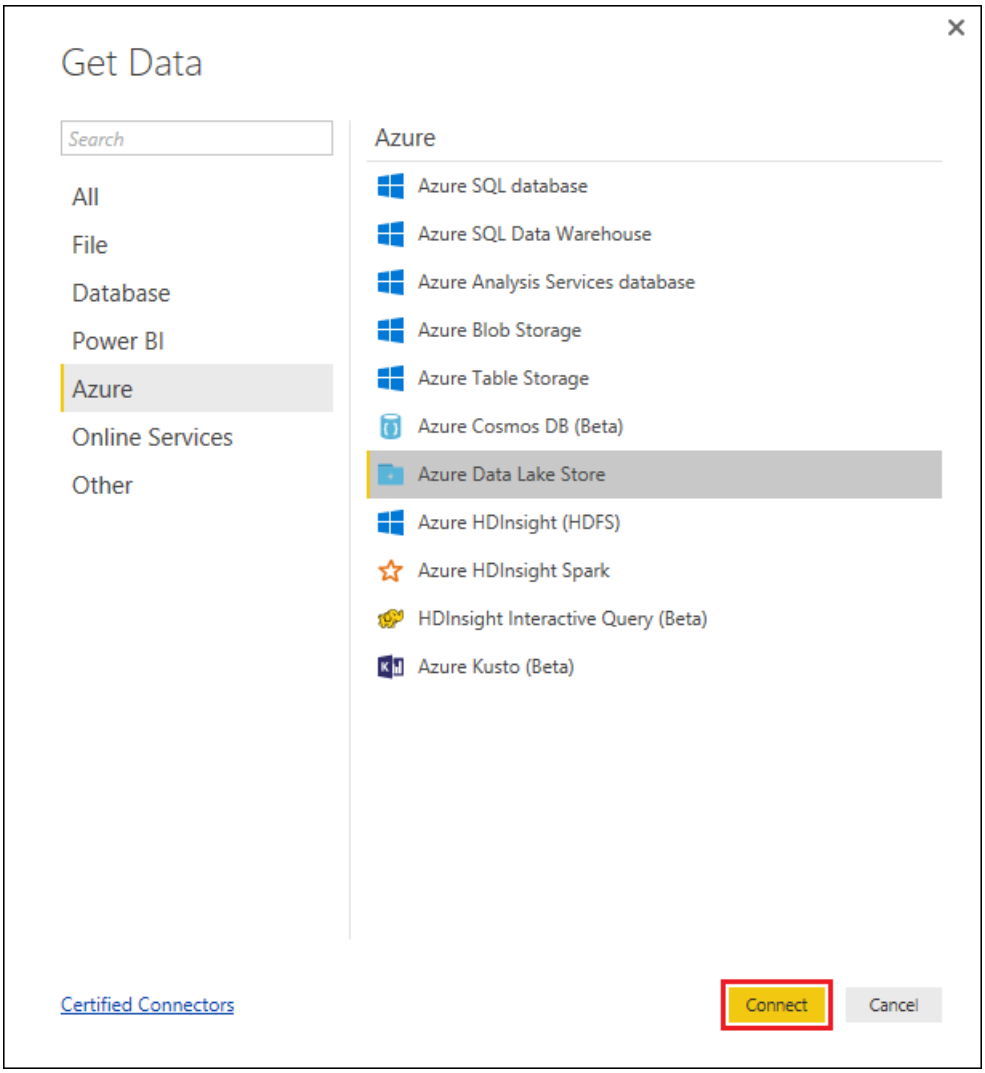
Prerequisites

Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **A Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal. This article assumes that you have already created a Data Lake Storage Gen1 account, called **myadlsg1**, and uploaded a sample data file (**Drivers.txt**) to it. This sample file is available for download from Azure Data Lake Git Repository.
- **Power BI Desktop**. You can download this from Microsoft Download Center.

Create a report in Power BI Desktop

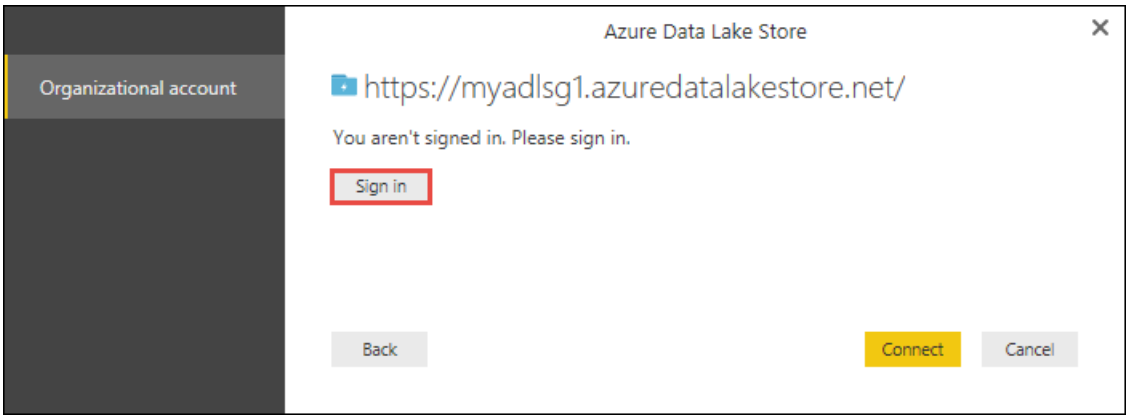
1. Launch Power BI Desktop on your computer.
2. From the **Home** ribbon, click **Get Data**, and then click More. In the **Get Data** dialog box, click **Azure**, click **Azure Data Lake Store**, and then click **Connect**.



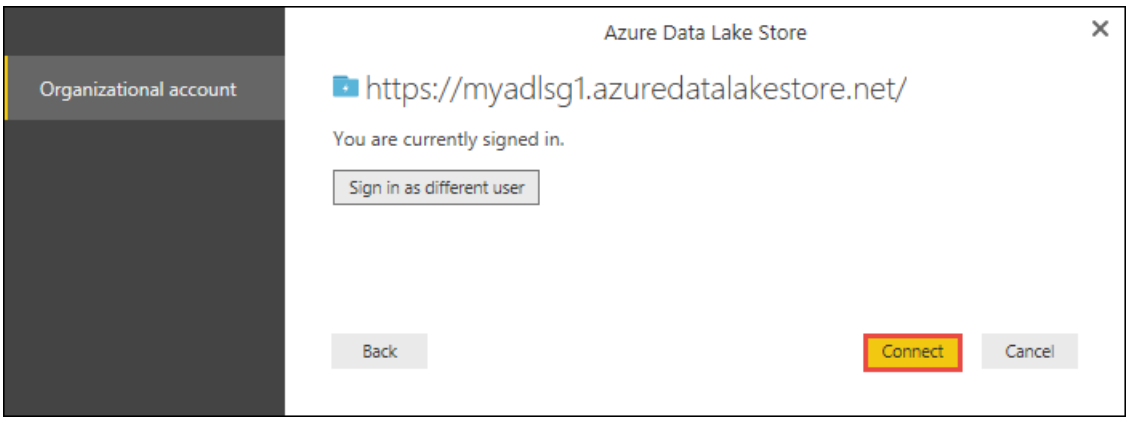
3. If you see a dialog box about the connector being in a development phase, opt to continue.

4. In the **Azure Data Lake Store** dialog box, provide the URL to your Data Lake Storage Gen1 account, and then click **OK**.

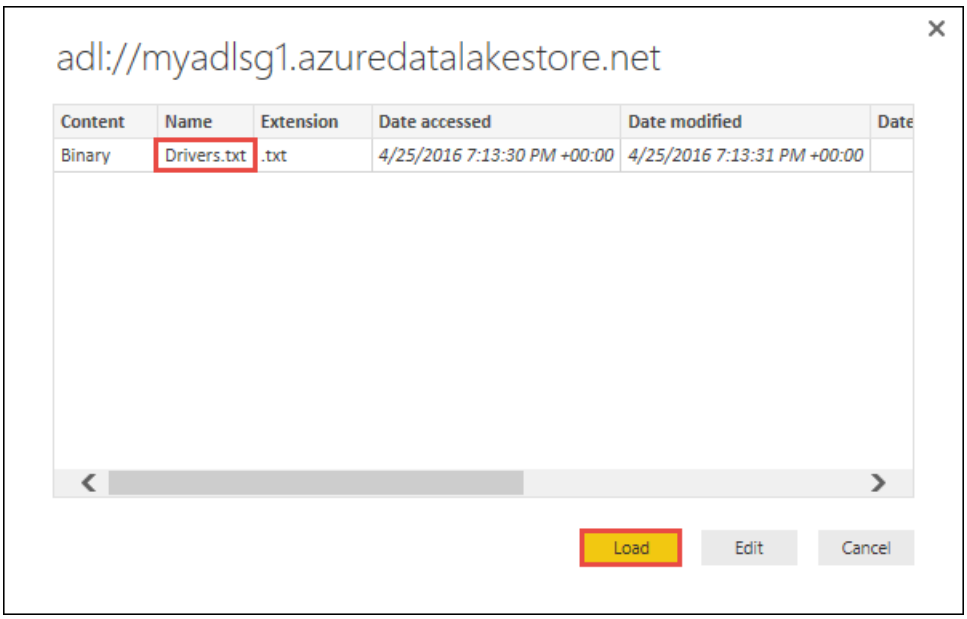
5. In the next dialog box, click **Sign in** to sign into the Data Lake Storage Gen1 account. You will be redirected to your organization's sign-in page. Follow the prompts to sign into the account.



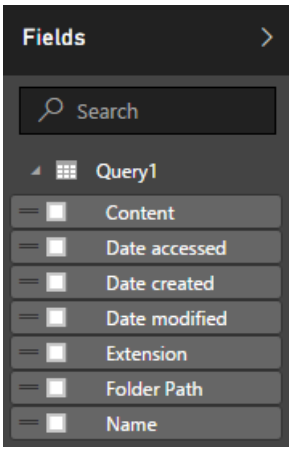
6. After you have successfully signed in, click **Connect**.



7. The next dialog box shows the file that you uploaded to your Data Lake Storage Gen1 account. Verify the info and then click **Load**.



8. After the data has been successfully loaded into Power BI, you will see the following fields in the **Fields** tab.

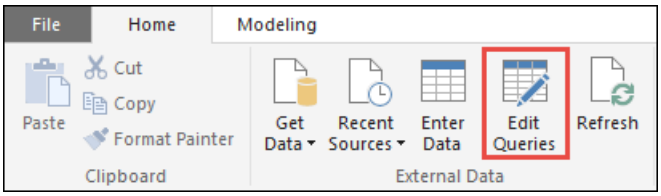


However, to visualize and analyze the data, we prefer the data to be available per the following fields

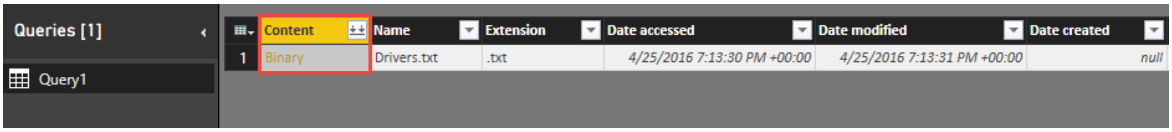
0	1	2	3	4	5	6
1	Maria Anders	Obere Str. 57	Berlin		12209	Germany
2	Ana Trujillo	Avda. de la Constitució...	México D.F.		5021	Mexico

In the next steps, we will update the query to convert the imported data in the desired format.

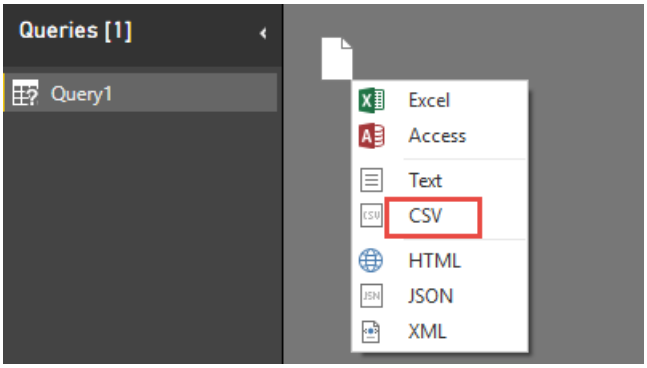
9. From the **Home** ribbon, click **Edit Queries**.



10. In the Query Editor, under the **Content** column, click **Binary**.

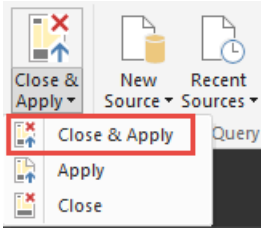


11. You will see a file icon, that represents the **Drivers.txt** file that you uploaded. Right-click the file, and click **CSV**.

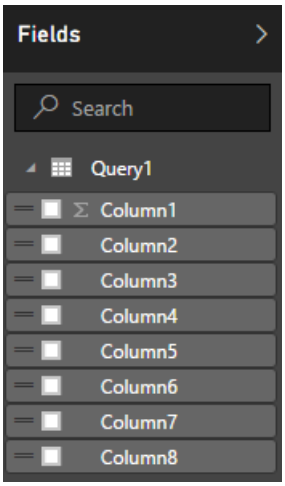


12. You should see an output as shown below. Your data is now available in a format that you can use to create visualizations.

13. From the **Home** ribbon, click **Close and Apply**, and then click **Close and Apply**.

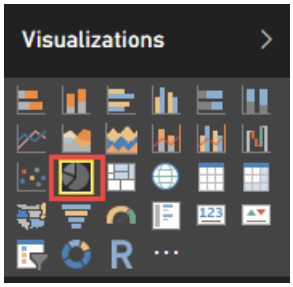


14. Once the query is updated, the **Fields** tab will show the new fields available for visualization.

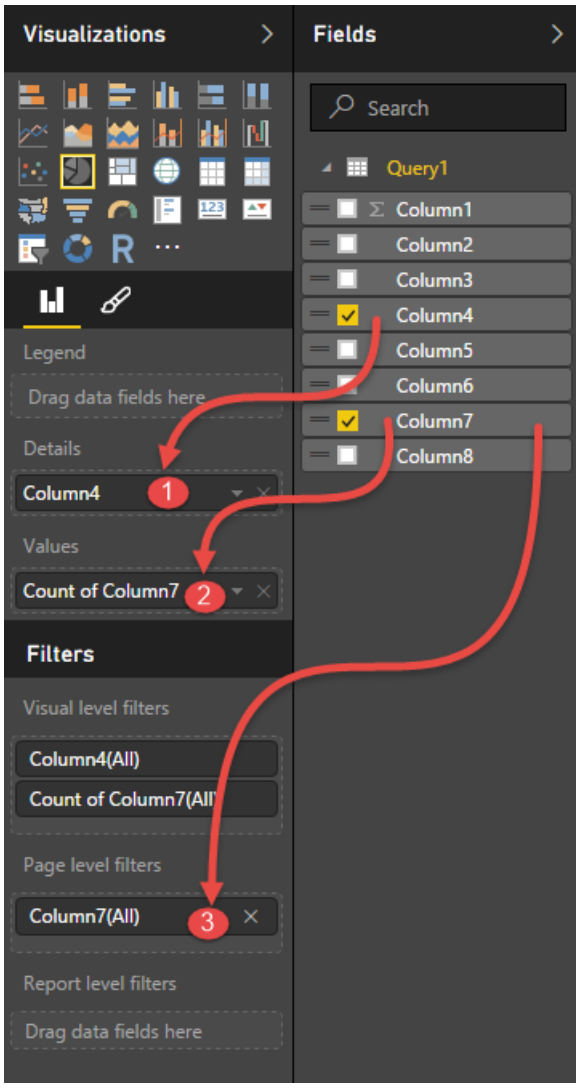


15. Let us create a pie chart to represent the drivers in each city for a given country/region. To do so, make the following selections.

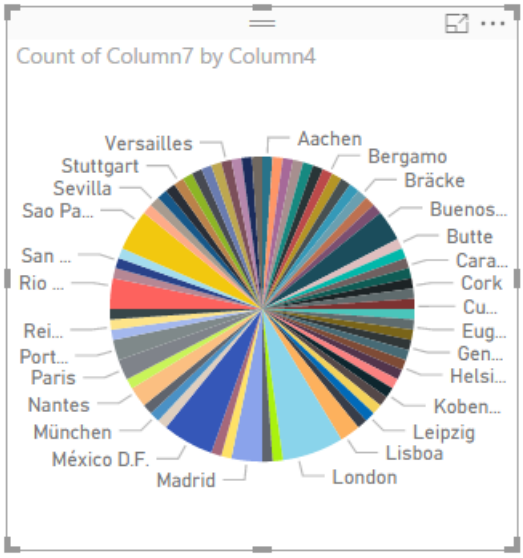
    a. From the Visualizations tab, click the symbol for a pie chart.



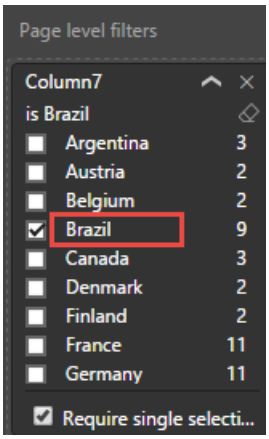
    b. The columns that we are going to use are **Column 4** (name of the city) and **Column 7** (name of the country/region). Drag these columns from **Fields** tab to **Visualizations** tab as shown below.



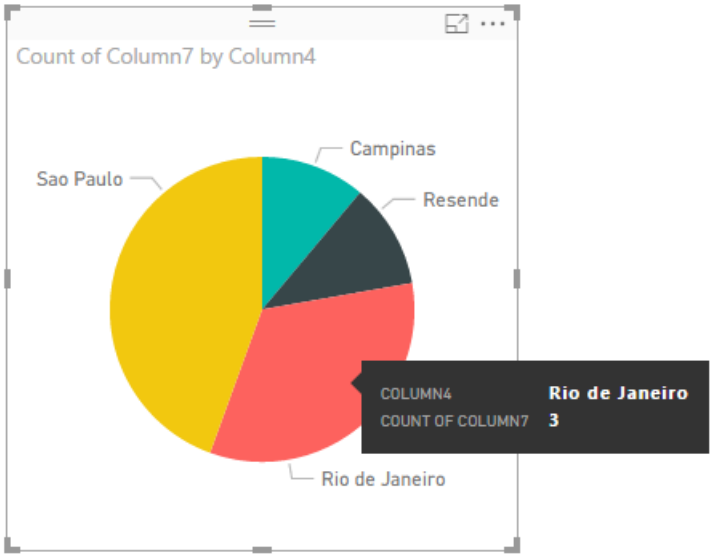
c. The pie chart should now resemble like the one shown below.



16. By selecting a specific country/region from the page level filters, you can now see the number of drivers in each city of the selected country/region. For example, under the **Visualizations** tab, under **Page level filters**, select **Brazil**.



17. The pie chart is automatically updated to display the drivers in the cities of Brazil.



18. From the **File** menu, click **Save** to save the visualization as a Power BI Desktop file.

Publish report to Power BI service

Once you have created the visualizations in Power BI Desktop, you can share it with others by publishing it to the Power BI service. For instructions on how to do that, see Publish from Power BI Desktop.

See also

- Analyze data in Data Lake Storage Gen1 using Data Lake Analytics

Register data from Azure Data Lake Storage Gen1 in Azure Data Catalog

10/3/2022 • 3 minutes to read • Edit Online

In this article, you will learn how to integrate Azure Data Lake Storage Gen1 with Azure Data Catalog to make your data discoverable within an organization by integrating it with Data Catalog. For more information on cataloging data, see Azure Data Catalog. To understand scenarios in which you can use Data Catalog, see Azure Data Catalog common scenarios.

Prerequisites

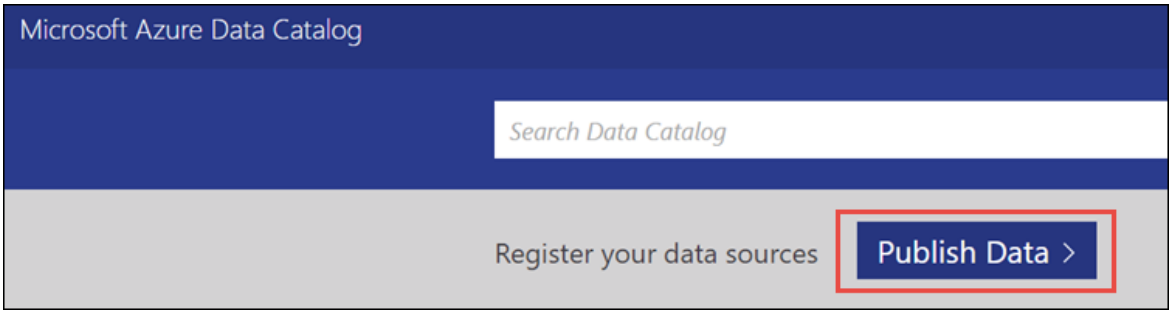
Before you begin this tutorial, you must have the following:

- **An Azure subscription**. See Get Azure free trial.
- **Enable your Azure subscription** for Data Lake Storage Gen1. See instructions.
- **A Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal. For this tutorial, create a Data Lake Storage Gen1 account called **datacatalogstore**.

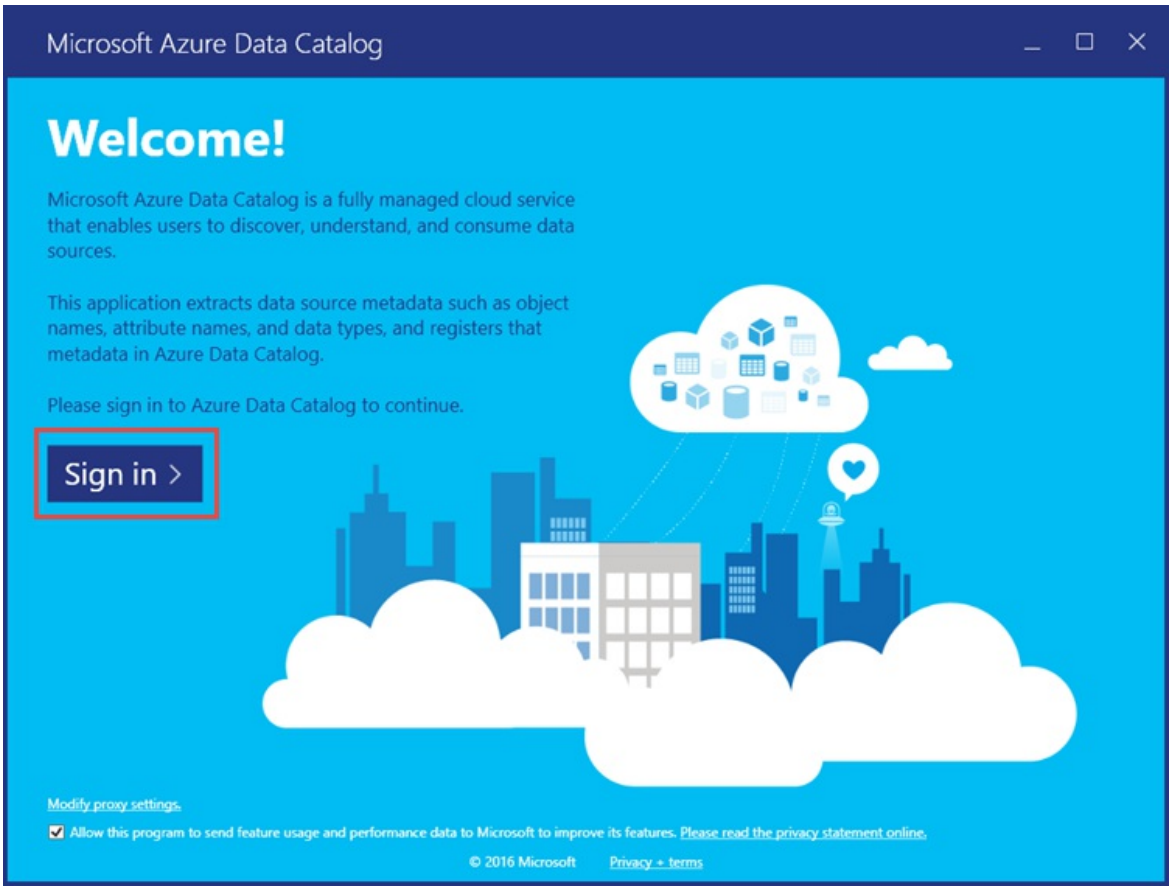
  Once you have created the account, upload a sample data set to it. For this tutorial, let us upload all the .csv files under the **AmbulanceData** folder in the Azure Data Lake Git Repository. You can use various clients, such as Azure Storage Explorer, to upload data to a blob container.
- **Azure Data Catalog**. Your organization must already have an Azure Data Catalog created for your organization. Only one catalog is allowed for each organization.

Register Data Lake Storage Gen1 as a source for Data Catalog

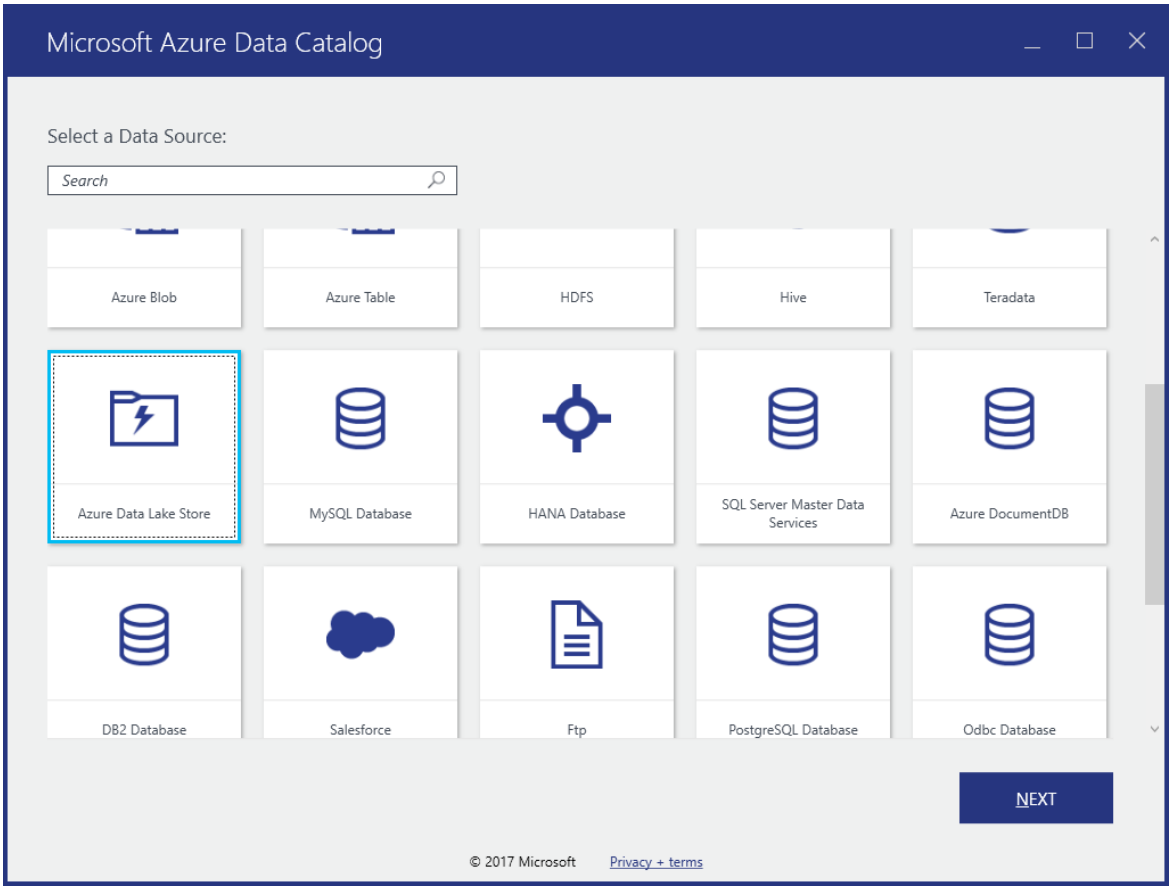
1. Go to `https://azure.microsoft.com/services/data-catalog`, and click **Get started**.
2. Log into the Azure Data Catalog portal, and click **Publish data**.



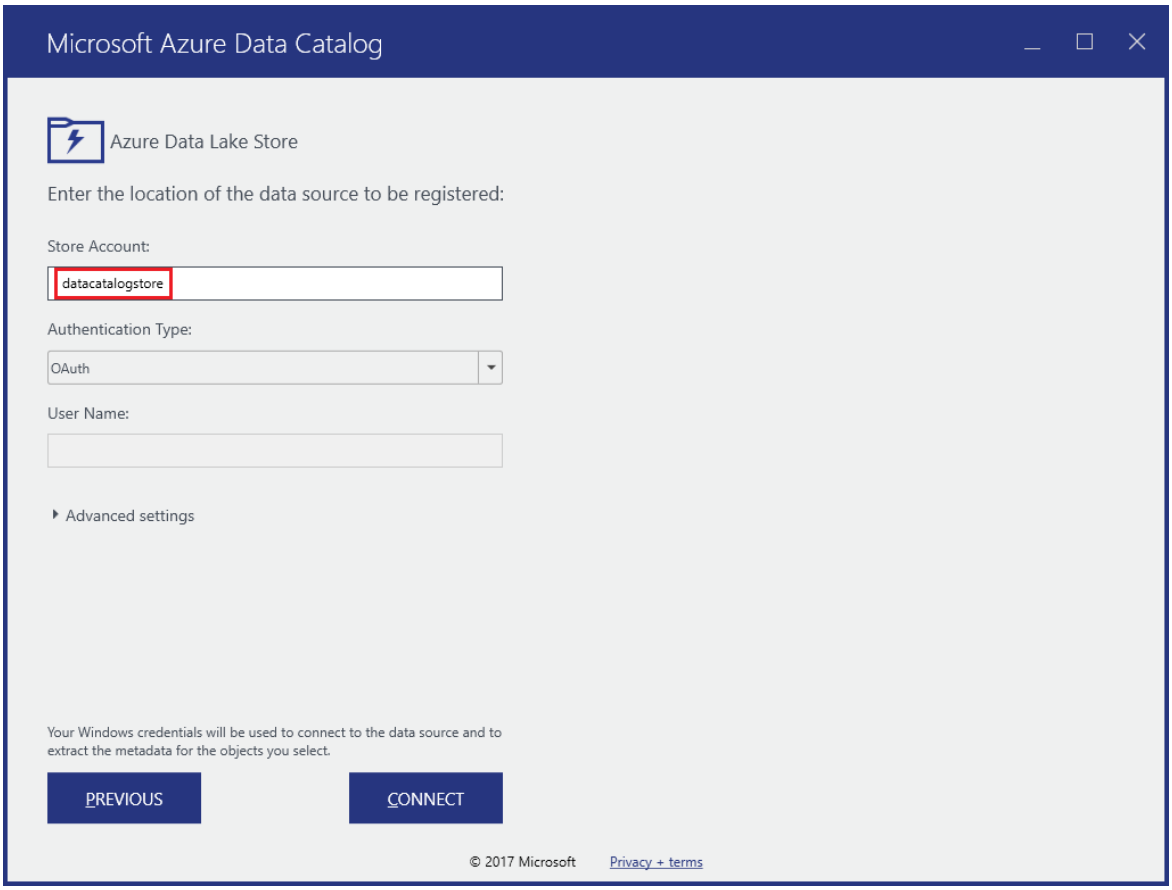
3. On the next page, click **Launch Application**. This will download the application manifest file on your computer. Double-click the manifest file to start the application.
4. On the Welcome page, click **Sign in**, and enter your credentials.



5. On the Select a Data Source page, select **Azure Data Lake Store**, and then click **Next**.



6. On the next page, provide the Data Lake Storage Gen1 account name that you want to register in Data Catalog. Leave the other options as default and then click **Connect**.

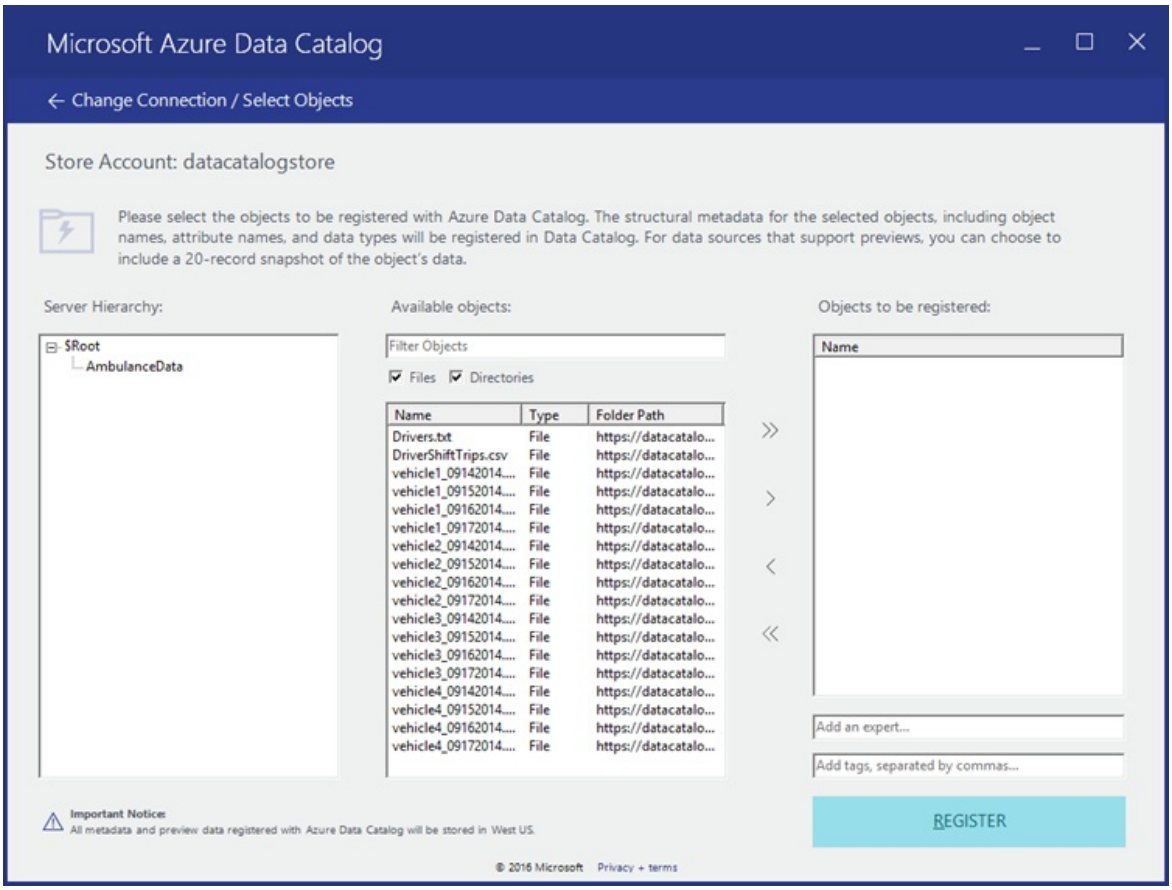


7. The next page can be divided into the following segments.

   a. The **Server Hierarchy** box represents the Data Lake Storage Gen1 account folder structure. **$Root** represents the Data Lake Storage Gen1 account root, and **AmbulanceData** represents the folder created in the root of the Data Lake Storage Gen1 account.

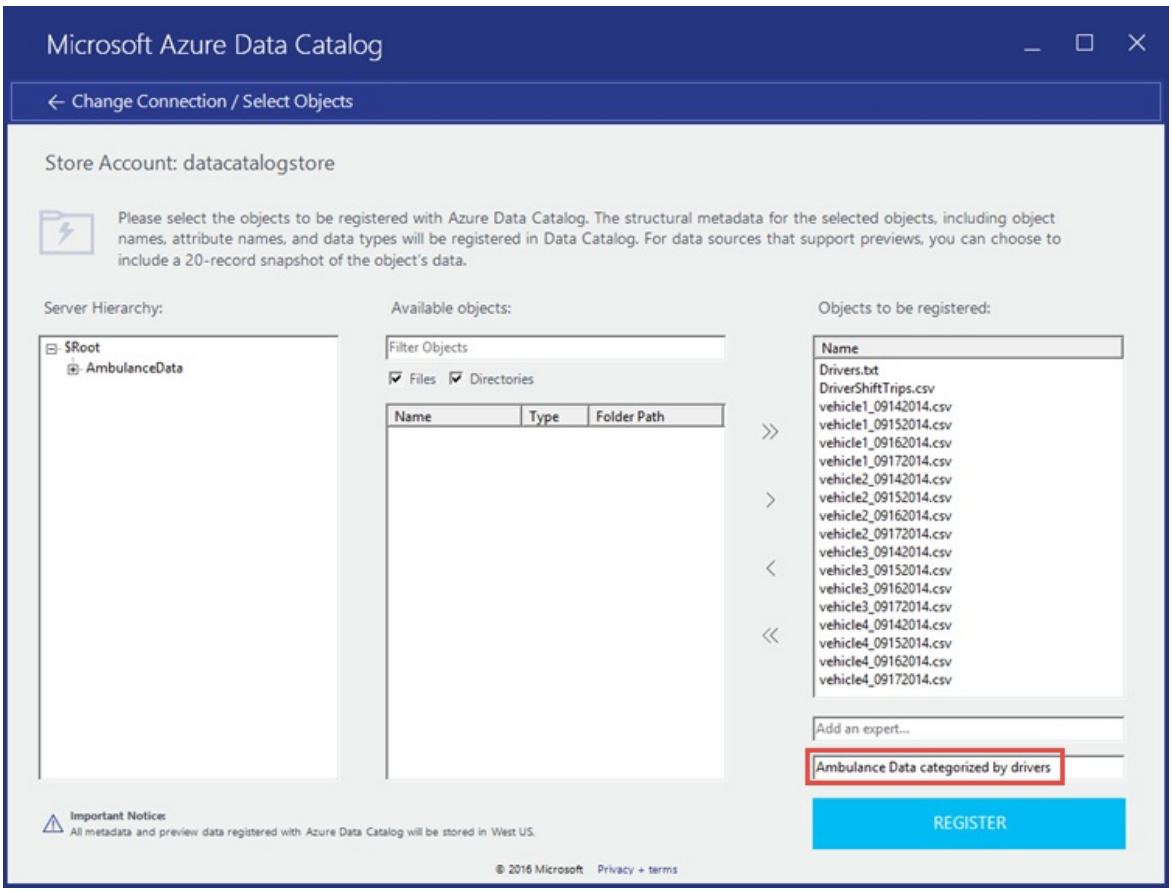
   b. The **Available objects** box lists the files and folders under the **AmbulanceData** folder.

   c. The **Objects to be registered** box lists the files and folders that you want to register in Azure Data Catalog.



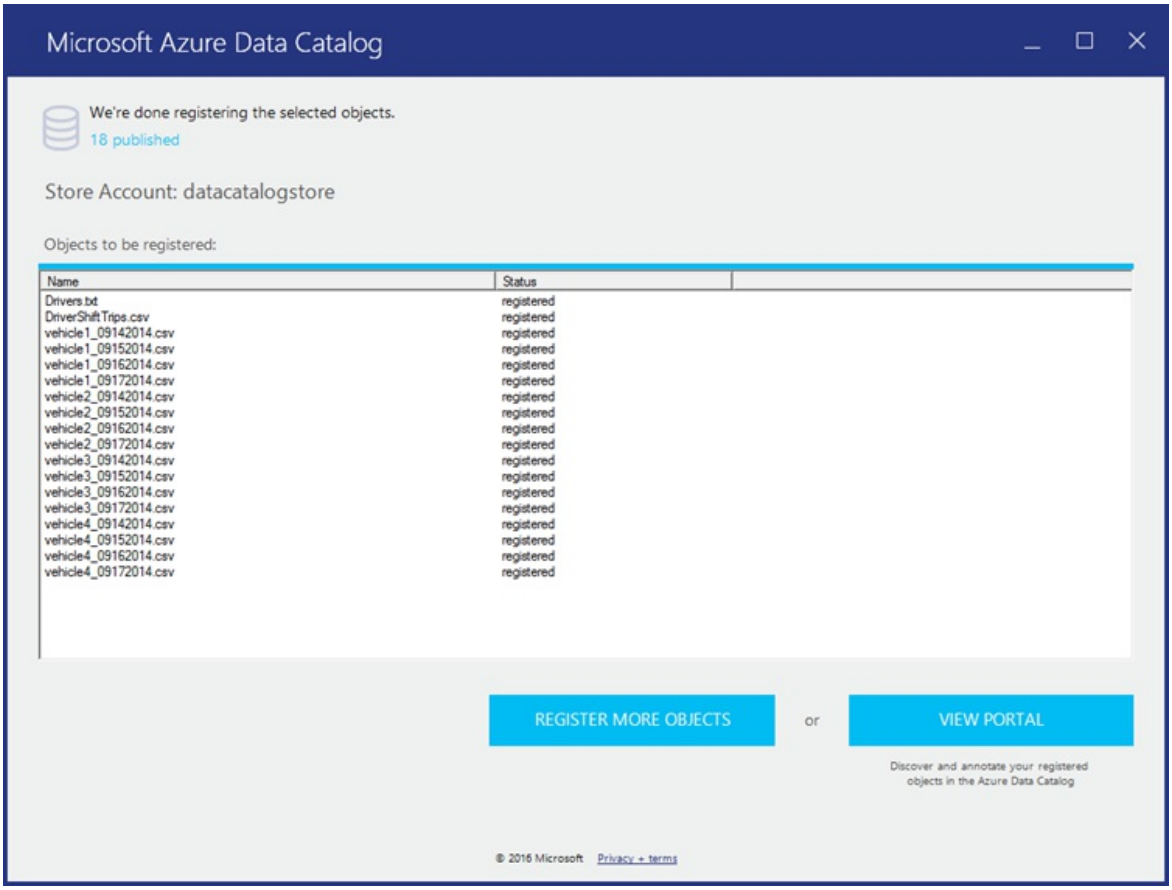
8. For this tutorial, you should register all the files in the directory. For that, click the () button to move all the files to **Objects to be registered** box.

   Because the data will be registered in an organization-wide data catalog, it is a recommended approach to add some metadata that you can later use to quickly locate the data. For example, you can add an e-mail address for the data owner (for example, one who is uploading the data) or add a tag to identify the data. The screen capture below shows a tag that you add to the data.

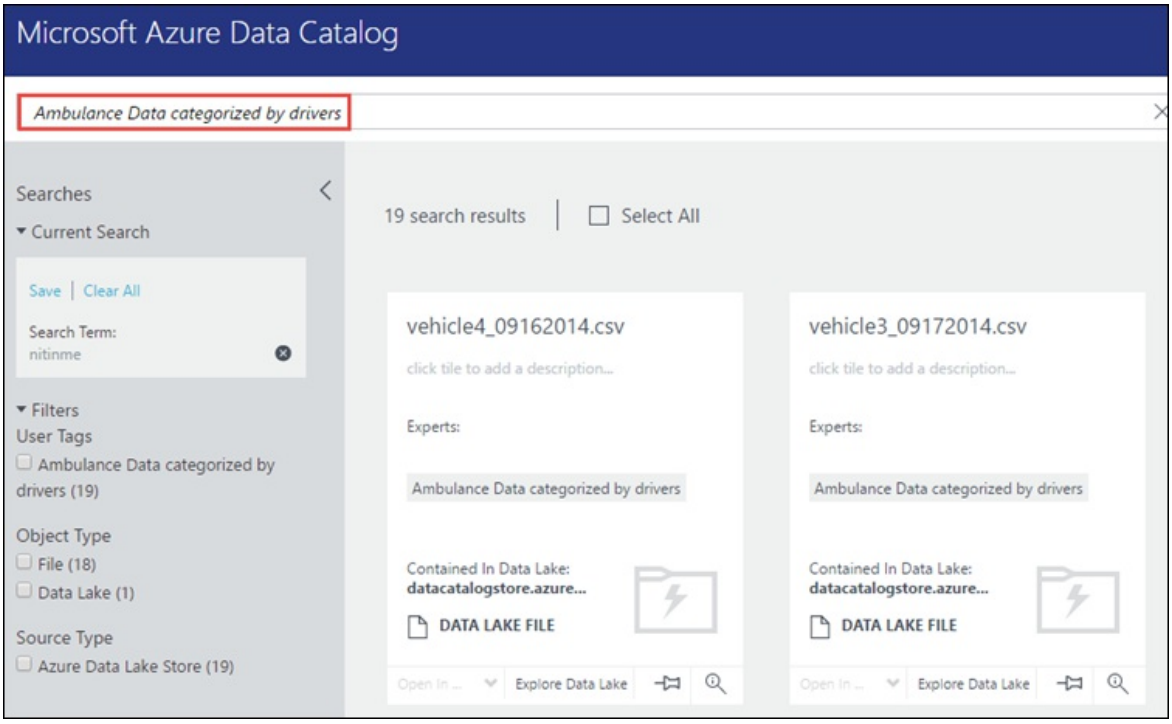


Click **Register**.

9. The following screen capture denotes that the data is successfully registered in the Data Catalog.



10. Click **View Portal** to go back to the Data Catalog portal and verify that you can now access the registered data from the portal. To search the data, you can use the tag you used while registering the data.



11. You can now perform operations like adding annotations and documentation to the data. For more information, see the following links.

- Annotate data sources in Data Catalog

- Document data sources in Data Catalog

See also

- Annotate data sources in Data Catalog
- Document data sources in Data Catalog
- Integrate Data Lake Storage Gen1 with other Azure services

Load data from Azure Data Lake Storage into dedicated SQL pools in Azure Synapse Analytics

10/3/2022 • 4 minutes to read • Edit Online

This guide outlines how to use the COPY statement to load data from Azure Data Lake Storage. For quick examples on using the COPY statement across all authentication methods, visit the following documentation: Securely load data using dedicated SQL pools.

> **NOTE**
>
> To provide feedback or report issues on the COPY statement, send an email to the following distribution list: sqldwcopypreview@service.microsoft.com.

- Create the target table to load data from Azure Data Lake Storage.
- Create the COPY statement to load data into the data warehouse.

If you don't have an Azure subscription, create a free account before you begin.

Before you begin

Before you begin this tutorial, download and install the newest version of SQL Server Management Studio (SSMS).

To run this tutorial, you need:

- A dedicated SQL pool. See Create a dedicated SQL pool and query data.
- A Data Lake Storage account. See Get started with Azure Data Lake Storage. For this storage account, you will need to configure or specify one of the following credentials to load: A storage account key, shared access signature (SAS) key, an Azure Directory Application user, or an Azure AD user that has the appropriate Azure role to the storage account.
- Currently, ingesting data using the COPY command into an Azure Storage account that is using the new Azure Storage DNS partition feature results in an error. Provision a storage account in a subscription that does not use DNS partitioning for this tutorial.

Create the target table

Connect to your dedicated SQL pool and create the target table you will load to. In this example, we are creating a product dimension table.

```
-- A: Create the target table
-- DimProduct
CREATE TABLE [dbo].[DimProduct]
(
    [ProductKey] [int] NOT NULL,
    [ProductLabel] [nvarchar](255) NULL,
    [ProductName] [nvarchar](500) NULL
)
WITH
(
    DISTRIBUTION = HASH([ProductKey]),
    CLUSTERED COLUMNSTORE INDEX
    --HEAP
);
```

Create the COPY statement

Connect to your SQL dedicated pool and run the COPY statement. For a complete list of examples, visit the following documentation: Securely load data using dedicated SQL pools.

```
-- B: Create and execute the COPY statement

COPY INTO [dbo].[DimProduct]
--The column list allows you map, omit, or reorder input file columns to target table columns.
--You can also specify the default value when there is a NULL value in the file.
--When the column list is not specified, columns will be mapped based on source and target ordinality
(
    ProductKey default -1 1,
    ProductLabel default 'myStringDefaultWhenNull' 2,
    ProductName default 'myStringDefaultWhenNull' 3
)
--The storage account location where you data is staged
FROM 'https://storageaccount.blob.core.windows.net/container/directory/'
WITH
(
   --CREDENTIAL: Specifies the authentication method and credential access your storage account
   CREDENTIAL = (IDENTITY = '', SECRET = ''),
   --FILE_TYPE: Specifies the file type in your storage account location
   FILE_TYPE = 'CSV',
   --FIELD_TERMINATOR: Marks the end of each field (column) in a delimited text (CSV) file
   FIELDTERMINATOR = '|',
   --ROWTERMINATOR: Marks the end of a record in the file
   ROWTERMINATOR = '0x0A',
   --FIELDQUOTE: Specifies the delimiter for data of type string in a delimited text (CSV) file
   FIELDQUOTE = '',
   ENCODING = 'UTF8',
   DATEFORMAT = 'ymd',
   --MAXERRORS: Maximum number of reject rows allowed in the load before the COPY operation is canceled
   MAXERRORS = 10,
   --ERRORFILE: Specifies the directory where the rejected rows and the corresponding error reason should be
written
   ERRORFILE = '/errorsfolder',
) OPTION (LABEL = 'COPY: ADLS tutorial');
```

Optimize columnstore compression

By default, tables are defined as a clustered columnstore index. After a load completes, some of the data rows might not be compressed into the columnstore. There's a variety of reasons why this can happen. To learn more, see manage columnstore indexes.

To optimize query performance and columnstore compression after a load, rebuild the table to force the

columnstore index to compress all the rows.

```
ALTER INDEX ALL ON [dbo].[DimProduct] REBUILD;
```

Optimize statistics

It is best to create single-column statistics immediately after a load. There are some choices for statistics. For example, if you create single-column statistics on every column it might take a long time to rebuild all the statistics. If you know certain columns are not going to be in query predicates, you can skip creating statistics on those columns.

If you decide to create single-column statistics on every column of every table, you can use the stored procedure code sample `prc_sqldw_create_stats` in the statistics article.

The following example is a good starting point for creating statistics. It creates single-column statistics on each column in the dimension table, and on each joining column in the fact tables. You can always add single or multi-column statistics to other fact table columns later on.

Achievement unlocked!

You have successfully loaded data into your data warehouse. Great job!

Next steps

Loading data is the first step to developing a data warehouse solution using Azure Synapse Analytics. Check out our development resources.

Learn how to develop tables for data warehousing

For more loading examples and references, view the following documentation:

- COPY statement reference documentation
- COPY examples for each authentication method
- COPY quickstart for a single table

Integrating Azure Data Lake Storage Gen1 with other Azure services

10/3/2022 • 3 minutes to read • Edit Online

Azure Data Lake Storage Gen1 can be used in conjunction with other Azure services to enable a wider range of scenarios. The following article lists the services that Data Lake Storage Gen1 can be integrated with.

Use Data Lake Storage Gen1 with Azure HDInsight

You can provision an Azure HDInsight cluster that uses Data Lake Storage Gen1 as the HDFS-compliant storage. For this release, for Hadoop and Storm clusters on Windows and Linux, you can use Data Lake Storage Gen1 only as an additional storage. Such clusters still use Azure Storage (WASB) as the default storage. However, for HBase clusters on Windows and Linux, you can use Data Lake Storage Gen1 as the default storage, or additional storage, or both.

For instructions on how to provision an HDInsight cluster with Data Lake Storage Gen1, see:

- Provision an HDInsight cluster with Data Lake Storage Gen1 using Azure portal
- Provision an HDInsight cluster with Data Lake Storage Gen1 as default storage using Azure PowerShell
- Provision an HDInsight cluster with Data Lake Storage Gen1 as additional storage using Azure PowerShell

Use Data Lake Storage Gen1 with Azure Data Lake Analytics

Azure Data Lake Analytics enables you to work with Big Data at cloud scale. It dynamically provisions resources and lets you do analytics on terabytes or even exabytes of data that can be stored in a number of supported data sources, one of them being Data Lake Storage Gen1. Data Lake Analytics is specially optimized to work with Data Lake Storage Gen1 - providing the highest level of performance, throughput, and parallelization for you big data workloads.

For instructions on how to use Data Lake Analytics with Data Lake Storage Gen1, see Get Started with Data Lake Analytics using Data Lake Storage Gen1.

Use Data Lake Storage Gen1 with Azure Data Factory

You can use Azure Data Factory to ingest data from Azure tables, Azure SQL Database, Azure SQL DataWarehouse, Azure Storage Blobs, and on-premises databases. Being a first class citizen in the Azure ecosystem, Azure Data Factory can be used to orchestrate the ingestion of data from these source to Data Lake Storage Gen1.

For instructions on how to use Azure Data Factory with Data Lake Storage Gen1, see Move data to and from Data Lake Storage Gen1 using Data Factory.

Copy data from Azure Storage Blobs into Data Lake Storage Gen1

Azure Data Lake Storage Gen1 provides a command-line tool, AdlCopy, that enables you to copy data from Azure Blob Storage into a Data Lake Storage Gen1 account. For more information, see Copy data from Azure Storage Blobs to Data Lake Storage Gen1.

Copy data between Azure SQL Database and Data Lake Storage Gen1

You can use Apache Sqoop to import and export data between Azure SQL Database and Data Lake Storage Gen1. For more information, see Copy data between Data Lake Storage Gen1 and Azure SQL Database using Sqoop.

Use Data Lake Storage Gen1 with Stream Analytics

You can use Data Lake Storage Gen1 as one of the outputs to store data streamed using Azure Stream Analytics. For more information, see Stream data from Azure Storage Blob into Data Lake Storage Gen1 using Azure Stream Analytics.

Use Data Lake Storage Gen1 with Power BI

You can use Power BI to import data from a Data Lake Storage Gen1 account to analyze and visualize the data. For more information, see Analyze data in Data Lake Storage Gen1 using Power BI.

Use Data Lake Storage Gen1 with Data Catalog

You can register data from Data Lake Storage Gen1 into the Azure Data Catalog to make the data discoverable throughout the organization. For more information see Register data from Data Lake Storage Gen1 in Azure Data Catalog.

Use Data Lake Storage Gen1 with SQL Server Integration Services (SSIS)

You can use the Data Lake Storage Gen1 connection manager in SSIS to connect an SSIS package with Data Lake Storage Gen1. For more information, see Use Data Lake Storage Gen1 with SSIS.

Use Data Lake Storage Gen1 with Azure Event Hubs

You can use Azure Data Lake Storage Gen1 to archive and capture data received by Azure Event Hubs. For more information see Use Data Lake Storage Gen1 with Azure Event Hubs.

See also

- Overview of Azure Data Lake Storage Gen1
- Get Started with Data Lake Storage Gen1 using Portal
- Get started with Data Lake Storage Gen1 using PowerShell

Accessing diagnostic logs for Azure Data Lake Storage Gen1

10/3/2022 • 6 minutes to read • Edit Online

Learn to enable diagnostic logging for your Azure Data Lake Storage Gen1 account and how to view the logs collected for your account.

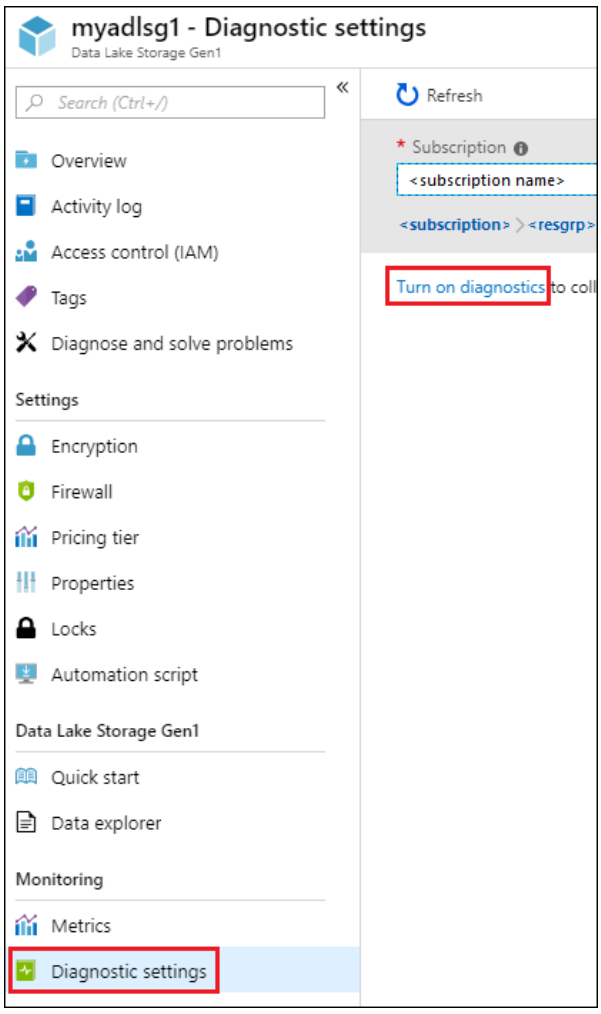
Organizations can enable diagnostic logging for their Azure Data Lake Storage Gen1 account to collect data access audit trails that provides information such as list of users accessing the data, how frequently the data is accessed, how much data is stored in the account, etc. When enabled, the diagnostics and/or requests are logged on a best-effort basis. Both Requests and Diagnostics log entries are created only if there are requests made against the service endpoint.

Prerequisites

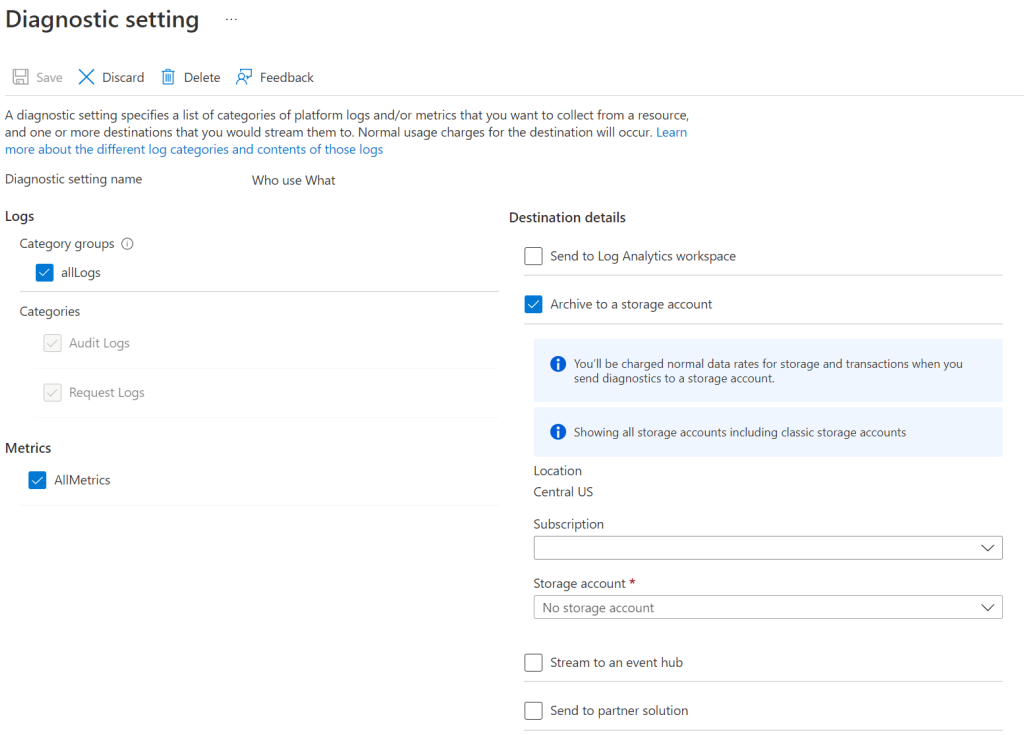
- **An Azure subscription**. See Get Azure free trial.
- **Azure Data Lake Storage Gen1 account**. Follow the instructions at Get started with Azure Data Lake Storage Gen1 using the Azure portal.

Enable diagnostic logging for your Data Lake Storage Gen1 account

1. Sign on to the new Azure portal.
2. Open your Data Lake Storage Gen1 account, and from your Data Lake Storage Gen1 account blade, click **Diagnostic settings**.
3. In the **Diagnostics settings** blade, click **Turn on diagnostics**.



4. In the **Diagnostics settings** blade, make the following changes to configure diagnostic logging.



- For **Name**, enter a value for the diagnostic log configuration.
- You can choose to store/process the data in different ways.
    - Select the option to **Archive to a storage account** to store logs to an Azure Storage account. You use this option if you want to archive the data that will be batch-processed at a later date. If you select this option you must provide an Azure Storage account to save the logs to.
    - Select the option to **Stream to an event hub** to stream log data to an Azure Event Hub. Most likely you will use this option if you have a downstream processing pipeline to analyze incoming logs at real time. If you select this option, you must provide the details for the Azure Event Hub you want to use.
    - Select the option to **Send to Log Analytics** to use the Azure Monitor service to analyze the generated log data. If you select this option, you must provide the details for the Log Analytics workspace that you would use the perform log analysis. See View or analyze data collected with Azure Monitor logs search for details on using Azure Monitor logs.
- Specify whether you want to get audit logs or request logs or both.
- Specify the number of days for which the data must be retained. Retention is only applicable if you are using Azure storage account to archive log data.
- Click **Save**.

Once you have enabled diagnostic settings, you can watch the logs in the **Diagnostic Logs** tab.

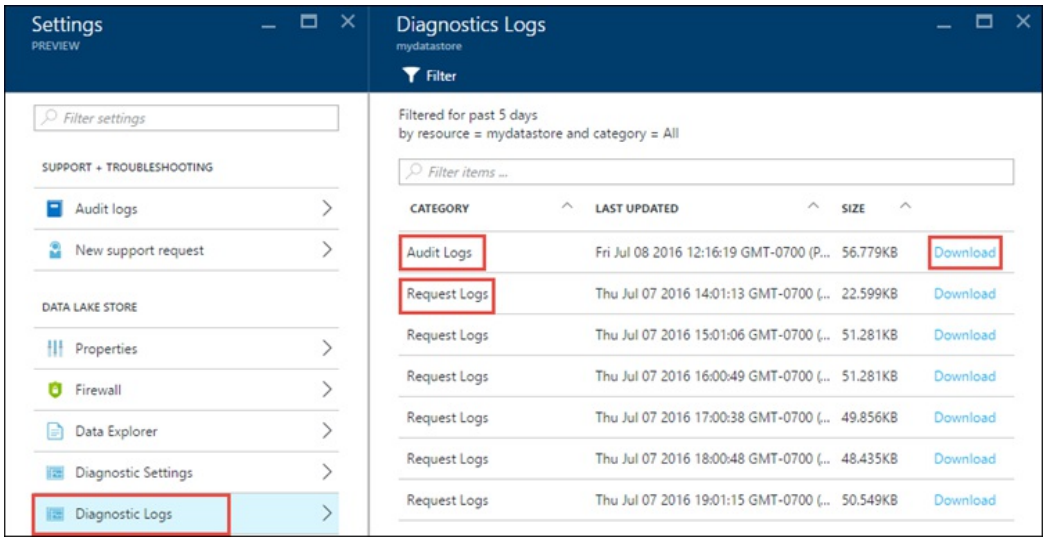
View diagnostic logs for your Data Lake Storage Gen1 account

There are two ways to view the log data for your Data Lake Storage Gen1 account.

- From the Data Lake Storage Gen1 account settings view
- From the Azure Storage account where the data is stored

Using the Data Lake Storage Gen1 Settings view

1. From your Data Lake Storage Gen1 account **Settings** blade, click **Diagnostic Logs**.



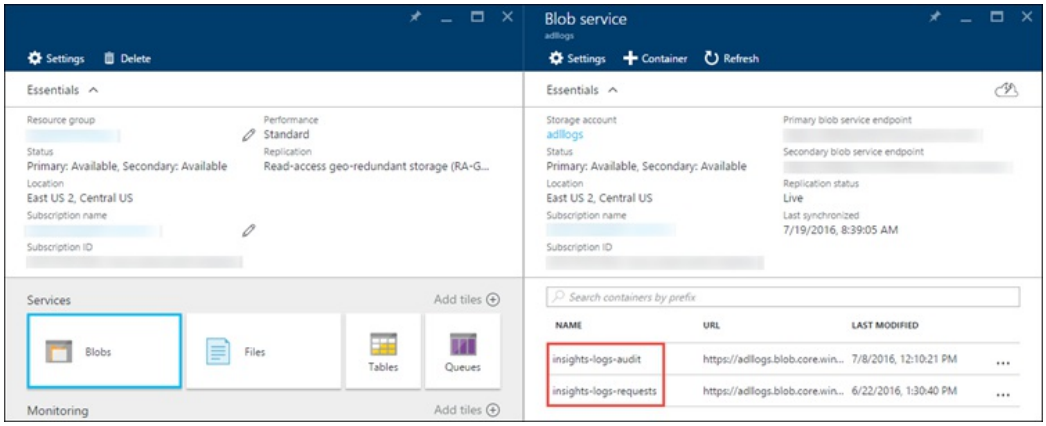
2. In the **Diagnostics Logs** blade, you should see the logs categorized by **Audit Logs** and **Request Logs**.

   - Request logs capture every API request made on the Data Lake Storage Gen1 account.
   - Audit Logs are similar to request Logs but provide a much more detailed breakdown of the operations being performed on the Data Lake Storage Gen1 account. For example, a single upload API call in request logs might result in multiple "Append" operations in the audit logs.

3. To download the logs, click the **Download** link against each log entry.

From the Azure Storage account that contains log data

1. Open the Azure Storage account blade associated with Data Lake Storage Gen1 for logging, and then click Blobs. The **Blob service** blade lists two containers.



   - The container **insights-logs-audit** contains the audit logs.
   - The container **insights-logs-requests** contains the request logs.

2. Within these containers, the logs are stored under the following structure.

As an example, the complete path to an audit log could be

```
https://adllogs.blob.core.windows.net/insights-logs-audit/resourceId=/SUBSCRIPTIONS/<sub-id>/RESOURCEGROUPS/myresourcegroup/PROVIDERS/MICROSOFT.DATALAKESTORE/ACCOUNTS/mydatalakestorage/y=2016/m=07/d=18/h=04/m=00/PT
```

Similarly, the complete path to a request log could be

```
https://adllogs.blob.core.windows.net/insights-logs-requests/resourceId=/SUBSCRIPTIONS/<sub-id>/RESOURCEGROUPS/myresourcegroup/PROVIDERS/MICROSOFT.DATALAKESTORE/ACCOUNTS/mydatalakestorage/y=2016/m=07/d=18/h=14/m=00/PT
```

Understand the structure of the log data

The audit and request logs are in a JSON format. In this section, we look at the structure of JSON for request and audit logs.

Request logs

Here's a sample entry in the JSON-formatted request log. Each blob has one root object called **records** that contains an array of log objects.

```
{
"records":
  [
    . . . .
    ,
    {
        "time": "2016-07-07T21:02:53.456Z",
        "resourceId":
"/SUBSCRIPTIONS/<subscription_id>/RESOURCEGROUPS/<resource_group_name>/PROVIDERS/MICROSOFT.DATALAKESTORE/ACCOUNTS/<data_lake_storage_gen1_account_name>",
        "category": "Requests",
        "operationName": "GETCustomerIngressEgress",
        "resultType": "200",
        "callerIpAddress": "::ffff:1.1.1.1",
        "correlationId": "4a11c709-05f5-417c-a98d-6e81b3e29c58",
        "identity": "1808bd5f-62af-45f4-89d8-03c5e81bac30",
        "properties":
{"HttpMethod":"GET","Path":"/webhdfs/v1/Samples/Outputs/Drivers.csv","RequestContentLength":0,"StoreIngressSize":0 ,"StoreEgressSize":4096,"ClientRequestId":"3b7adbd9-3519-4f28-a61c-bd89506163b8","StartTime":"2016-07-07T21:02:52.472Z","EndTime":"2016-07-07T21:02:53.456Z","QueryParameters":"api-version=<version>&op=<operationName>"}
    }
    ,
    . . . .
  ]
}
```

Request log schema

NAME	TYPE	DESCRIPTION
time	String	The timestamp (in UTC) of the log
resourceId	String	The ID of the resource that operation took place on
category	String	The log category. For example, **Requests**.
operationName	String	Name of the operation that is logged. For example, getfilestatus.
resultType	String	The status of the operation, For example, 200.
callerIpAddress	String	The IP address of the client making the request
correlationId	String	The ID of the log that can used to group together a set of related log entries
identity	Object	The identity that generated the log
properties	JSON	See below for details

Request log properties schema

NAME	TYPE	DESCRIPTION
HttpMethod	String	The HTTP Method used for the operation. For example, GET.
Path	String	The path the operation was performed on
RequestContentLength	int	The content length of the HTTP request
ClientRequestId	String	The ID that uniquely identifies this request
StartTime	String	The time at which the server received the request
EndTime	String	The time at which the server sent a response
StoreIngressSize	Long	Size in bytes ingressed to Data Lake Store
StoreEgressSize	Long	Size in bytes egressed from Data Lake Store

NAME	TYPE	DESCRIPTION
QueryParameters	String	Description: These are the http query parameters. Example 1: api-version=2014-01-01&op=getfilestatus Example 2: op=APPEND&append=true&syncFlag=DATA&filesessionid=bee3355a-4925-4435-bb4d-ceea52811aeb&leaseid=bee3355a-4925-4435-bb4d-ceea52811aeb&offset=28313319&api-version=2017-08-01

Audit logs

Here's a sample entry in the JSON-formatted audit log. Each blob has one root object called **records** that contains an array of log objects

```
{
"records":
  [
    . . . .
    ,
    {
        "time": "2016-07-08T19:08:59.359Z",
        "resourceId":
"/SUBSCRIPTIONS/<subscription_id>/RESOURCEGROUPS/<resource_group_name>/PROVIDERS/MICROSOFT.DATALAKESTORE/ACC
OUNTS/<data_lake_storage_gen1_account_name>",
        "category": "Audit",
        "operationName": "SeOpenStream",
        "resultType": "0",
        "resultSignature": "0",
        "correlationId": "381110fc03534e1cb99ec52376ceebdf;Append_BrEKAmg;25.66.9.145",
        "identity": "A9DAFFAF-FFEE-4BB5-A4A0-1B6CBBF24355",
        "properties":
{"StreamName":"adl://<data_lake_storage_gen1_account_name>.azuredatalakestore.net/logs.csv"}
    }
    ,
    . . . .
  ]
}
```

Audit log schema

NAME	TYPE	DESCRIPTION
time	String	The timestamp (in UTC) of the log
resourceId	String	The ID of the resource that operation took place on
category	String	The log category. For example, **Audit**.
operationName	String	Name of the operation that is logged. For example, getfilestatus.
resultType	String	The status of the operation, For example, 200.
resultSignature	String	Additional details on the operation.
correlationId	String	The ID of the log that can used to group together a set of related log entries
identity	Object	The identity that generated the log

NAME	TYPE	DESCRIPTION
properties	JSON	See below for details

Audit log properties schema

NAME	TYPE	DESCRIPTION
StreamName	String	The path the operation was performed on

Samples to process the log data

When sending logs from Azure Data Lake Storage Gen1 to Azure Monitor logs (see View or analyze data collected with Azure Monitor logs search for details on using Azure Monitor logs), the following query will return a table containing a list of user display names, the time of the events, and the count of events for the time of the event along with a visual chart. It can easily be modified to show user GUID or other attributes:

```
search *
| where ( Type == "AzureDiagnostics" )
| summarize count(TimeGenerated) by identity_s, TimeGenerated
```

Azure Data Lake Storage Gen1 provides a sample on how to process and analyze the log data. You can find the sample at https://github.com/Azure/AzureDataLake/tree/master/Samples/AzureDiagnosticsSample.

See also

- Overview of Azure Data Lake Storage Gen1
- Secure data in Data Lake Storage Gen1

High availability and disaster recovery guidance for Data Lake Storage Gen1

10/3/2022 • 2 minutes to read • Edit Online

Data Lake Storage Gen1 provides locally redundant storage (LRS). Therefore, the data in your Data Lake Storage Gen1 account is resilient to transient hardware failures within a datacenter through automated replicas. This ensures durability and high availability, meeting the Data Lake Storage Gen1 SLA. This article provides guidance on how to further protect your data from rare region-wide outages or accidental deletions.

Disaster recovery guidance

It's critical for you to prepare a disaster recovery plan. Review the information in this article and these additional resources to help you create your own plan.

- Disaster recovery and high availability for Azure applications
- Azure resiliency technical guidance

Best practice recommendations

We recommend that you copy your critical data to another Data Lake Storage Gen1 account in another region with a frequency aligned to the needs of your disaster recovery plan. There are a variety of methods to copy data including ADLCopy, Azure PowerShell, or Azure Data Factory. Azure Data Factory is a useful service for creating and deploying data movement pipelines on a recurring basis.

If a regional outage occurs, you can then access your data in the region where the data was copied. You can monitor the Azure Service Health Dashboard to determine the Azure service status across the globe.

Data corruption or accidental deletion recovery guidance

While Data Lake Storage Gen1 provides data resiliency through automated replicas, this does not prevent your application (or developers/users) from corrupting data or accidentally deleting it.

To prevent accidental deletion, we recommend that you first set the correct access policies for your Data Lake Storage Gen1 account. This includes applying Azure resource locks to lock down important resources and applying account and file level access control using the available Data Lake Storage Gen1 security features. We also recommend that you routinely create copies of your critical data using ADLCopy, Azure PowerShell or Azure Data Factory in another Data Lake Storage Gen1 account, folder, or Azure subscription. This can be used to recover from a data corruption or deletion incident. Azure Data Factory is a useful service for creating and deploying data movement pipelines on a recurring basis.

You can also enable diagnostic logging for a Data Lake Storage Gen1 account to collect data access audit trails. The audit trails provide information about who might have deleted or updated a file.

Next steps

- Get started with Data Lake Storage Gen1
- Secure data in Data Lake Storage Gen1

Azure Policy built-in definitions for Azure Data Lake Storage Gen1

10/3/2022 • 2 minutes to read • Edit Online

This page is an index of Azure Policy built-in policy definitions for Azure Data Lake Storage Gen1. For additional Azure Policy built-ins for other services, see Azure Policy built-in definitions.

The name of each built-in policy definition links to the policy definition in the Azure portal. Use the link in the **Version** column to view the source on the Azure Policy GitHub repo.

Azure Data Lake Storage Gen1

NAME (AZURE PORTAL)	DESCRIPTION	EFFECT(S)	VERSION (GITHUB)
Deploy Diagnostic Settings for Data Lake Storage Gen1 to Event Hub	Deploys the diagnostic settings for Data Lake Storage Gen1 to stream to a regional Event Hub when any Data Lake Storage Gen1 which is missing this diagnostic settings is created or updated.	DeployIfNotExists, Disabled	2.0.0
Deploy Diagnostic Settings for Data Lake Storage Gen1 to Log Analytics workspace	Deploys the diagnostic settings for Data Lake Storage Gen1 to stream to a regional Log Analytics workspace when any Data Lake Storage Gen1 which is missing this diagnostic settings is created or updated.	DeployIfNotExists, Disabled	1.0.0
Require encryption on Data Lake Store accounts	This policy ensures encryption is enabled on all Data Lake Store accounts	deny	1.0.0
Resource logs in Azure Data Lake Store should be enabled	Audit enabling of resource logs. This enables you to recreate activity trails to use for investigation purposes; when a security incident occurs or when your network is compromised	AuditIfNotExists, Disabled	5.0.0

Next steps

- See the built-ins on the Azure Policy GitHub repo.
- Review the Azure Policy definition structure.
- Review Understanding policy effects.