**Introduction**

Completed100 XP

* 1 minute

Most software applications need to store data. Often this takes the form of a relational database, in which the data is organized in related tables and managed by using Structured Query Language (SQL). However, many applications don't need the rigid structure of a relational database and rely on non-relational (often referred to as NoSQL) storage.

Azure Storage is one of the core services in Microsoft Azure, and offers a range of options for storing data in the cloud. In this module, you'll explore the fundamental capabilities of Azure storage and learn how it's used to support applications that require non-relational data stores.

**Learning objectives**

In this module, you'll learn how to:

* Describe features and capabilities of Azure blob storage
* Describe features and capabilities of Azure Data Lake Gen2
* Describe features and capabilities of Azure file storage
* Describe features and capabilities of Azure table storage
* Provision and use an Azure Storage account

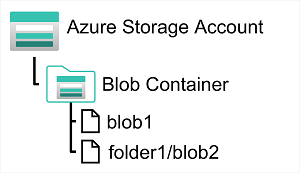
**Next unit: Explore Azure blob storage**

**Explore Azure blob storage**

Completed100 XP

* 4 minutes

Azure Blob Storage is a service that enables you to store massive amounts of unstructured data as binary large objects, or *blobs*, in the cloud. Blobs are an efficient way to store data files in a format that is optimized for cloud-based storage, and applications can read and write them by using the Azure blob storage API.



In an Azure storage account, you store blobs in *containers*. A container provides a convenient way of grouping related blobs together. You control who can read and write blobs inside a container at the container level.

Within a container, you can organize blobs in a hierarchy of virtual folders, similar to files in a file system on disk. However, by default, these folders are simply a way of using a "/" character in a blob name to organize the blobs into namespaces. The folders are purely virtual, and you can't perform folder-level operations to control access or perform bulk operations.

Azure Blob Storage supports three different types of blob:

* **Block blobs**. A block blob is handled as a set of blocks. Each block can vary in size, up to 100 MB. A block blob can contain up to 50,000 blocks, giving a maximum size of over 4.7 TB. The block is the smallest amount of data that can be read or written as an individual unit. Block blobs are best used to store discrete, large, binary objects that change infrequently.
* **Page blobs**. A page blob is organized as a collection of fixed size 512-byte pages. A page blob is optimized to support random read and write operations; you can fetch and store data for a single page if necessary. A page blob can hold up to 8 TB of data. Azure uses page blobs to implement virtual disk storage for virtual machines.
* **Append blobs**. An append blob is a block blob optimized to support append operations. You can only add blocks to the end of an append blob; updating or deleting existing blocks isn't supported. Each block can vary in size, up to 4 MB. The maximum size of an append blob is just over 195 GB.

Blob storage provides three access tiers, which help to balance access latency and storage cost:

* The *Hot* tier is the default. You use this tier for blobs that are accessed frequently. The blob data is stored on high-performance media.
* The *Cool* tier has lower performance and incurs reduced storage charges compared to the Hot tier. Use the Cool tier for data that is accessed infrequently. It's common for newly created blobs to be accessed frequently initially, but less so as time passes. In these situations, you can create the blob in the Hot tier, but migrate it to the Cool tier later. You can migrate a blob from the Cool tier back to the Hot tier.
* The *Archive* tier provides the lowest storage cost, but with increased latency. The Archive tier is intended for historical data that mustn't be lost, but is required only rarely. Blobs in the Archive tier are effectively stored in an offline state. Typical reading latency for the Hot and Cool tiers is a few milliseconds, but for the Archive tier, it can take hours for the data to become available. To retrieve a blob from the Archive tier, you must change the access tier to Hot or Cool. The blob will then be rehydrated. You can read the blob only when the rehydration process is complete.

You can create lifecycle management policies for blobs in a storage account. A lifecycle management policy can automatically move a blob from Hot to Cool, and then to the Archive tier, as it ages and is used less frequently (policy is based on the number of days since modification). A lifecycle management policy can also arrange to delete outdated blobs.

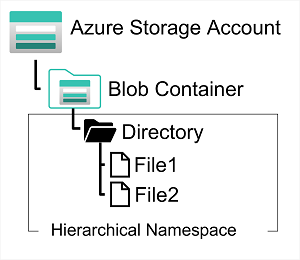
**Next unit: Explore Azure DataLake Storage Gen2**

**Explore Azure DataLake Storage Gen2**

Completed100 XP

* 3 minutes

Azure Data Lake Store (Gen1) is a separate service for hierarchical data storage for analytical data lakes, often used by so-called *big data* analytical solutions that work with structured, semi-structured, and unstructured data stored in files. Azure Data Lake Storage Gen**2** is a newer version of this service that is integrated into Azure Storage; enabling you to take advantage of the scalability of blob storage and the cost-control of storage tiers, combined with the hierarchical file system capabilities and compatibility with major analytics systems of Azure Data Lake Store.



Systems like Hadoop in Azure HDInsight, Azure Databricks, and Azure Synapse Analytics can mount a distributed file system hosted in Azure Data Lake Store Gen2 and use it to process huge volumes of data.

To create an Azure Data Lake Store Gen2 files system, you must enable the **Hierarchical Namespace** option of an Azure Storage account. You can do this when initially creating the storage account, or you can upgrade an existing Azure Storage account to support Data Lake Gen2. Be aware however that upgrading is a one-way process – after upgrading a storage account to support a hierarchical namespace for blob storage, you can’t revert it to a flat namespace.

**Next unit: Explore Azure Files**