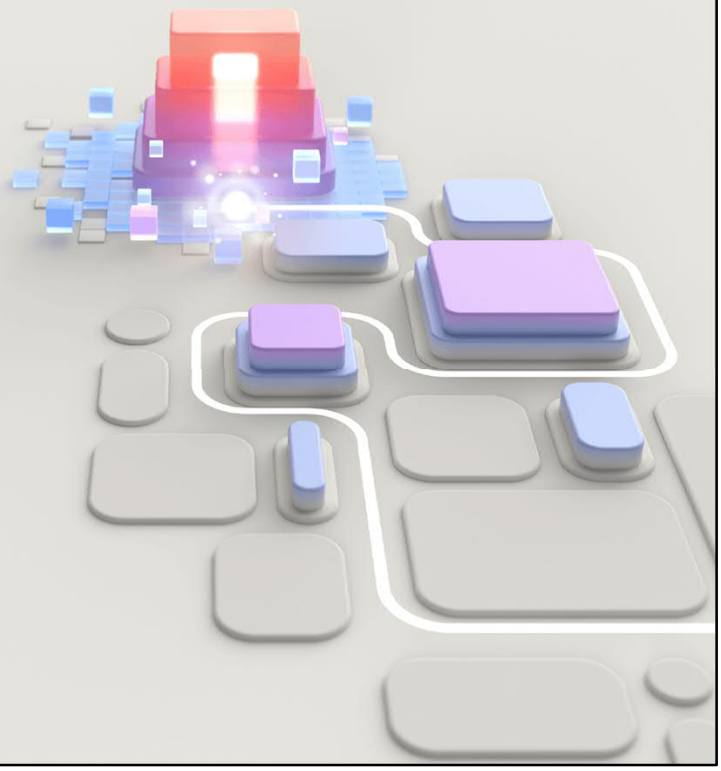
Microsoft

Explore fundamentals of non-relational data in Azure



Agenda

- Fundamentals of Azure Storage
- Fundamentals of Azure Cosmos DB

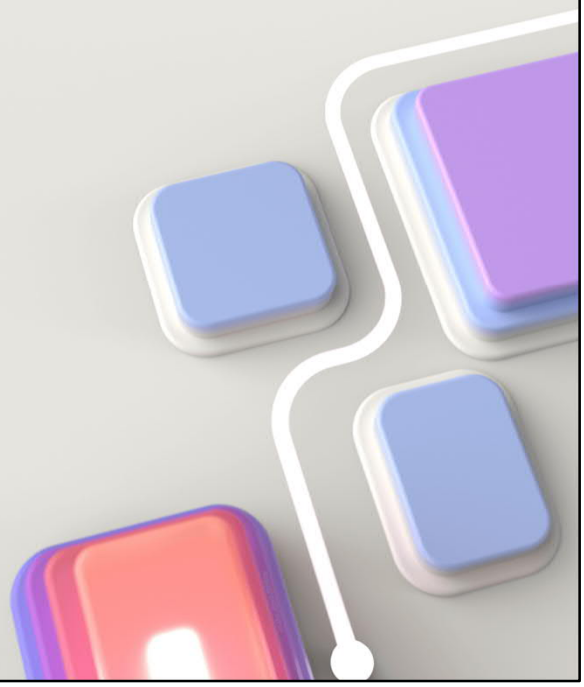
This should take approximately 90 minutes to deliver, including 15-20 minutes for each lab exercise.

Learning objectives

After completing this module, you will be able to:

1. Describe features and capabilities of Azure blob storage, Azure Data Lake Gen2, Azure file storage, and Azure table storage.
2. Provision and use an Azure Storage account.
3. Describe key features and capabilities of Azure Cosmos DB.
4. Identify the APIs supported in Azure Cosmos DB.
5. Provision and use an Azure Cosmos DB instance.

1: Fundamentals of Azure Storage



Azure Blob Storage

Storage for data as binary large objects (BLOBs)

Block blobs

- Large, discrete, binary objects that change infrequently
- Blobs can be up to 4.7 TB, composed of blocks of up to 100 MB
  - A blob can contain up to 50,000 blocks

Page blobs

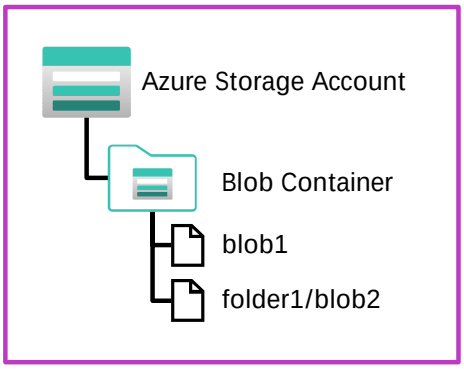
- Used as virtual disk storage for VMs
- Blobs can be up to 8 TB, composed of fixed sized-512 byte pages

Append blobs

- Block blobs that are used to optimize append operations
- Maximum size just over 195 GB – each block can be up to 4 MB

Per-blob storage tiers

- Hot – Highest cost, lowest latency
- Cool – Lower cost, higher latency
- Archive – Lowest cost, highest latency



Blobs can be organized in virtual directories, but each path is considered a single blob in a flat namespace – folder level operations are not supported

➢ **animated slide**

Azure Blob Storage is a service that enables you to store massive amounts of unstructured data, or *blobs*, in the cloud. Inside an Azure storage account, you create blobs inside *containers*. A container provides a convenient way of grouping related blobs together, and you can organize blobs in a hierarchy of folders, similar to files in a file system on disk. You control who can read and write blobs inside a container at the container level.

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.

>click to initiate animation

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. This 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. This 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.

>click to initiate animation

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.

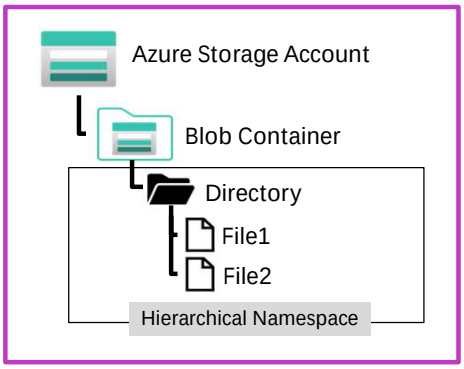
Azure Data Lake Store Gen 2

Distributed file system built on Blob Storage

- Combines Azure Data Lake Store Gen 1 with Azure Blob Storage for large-scale file storage and analytics
- Enables file and directory level access control and management
- Compatible with common large scale analytical systems

Enabled in an Azure Storage account through the *Hierarchical Namespace* option

- Set during account creation
- Upgrade existing storage account
  - One-way upgrade process



File system includes directories and files, and is compatible with large scale data analytics systems like Databricks

Azure Data Lake Storage Gen 1 is a separate service for hierarchical data storage for analytical data lakes. Azure Data Lake Storage Gen 2 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.

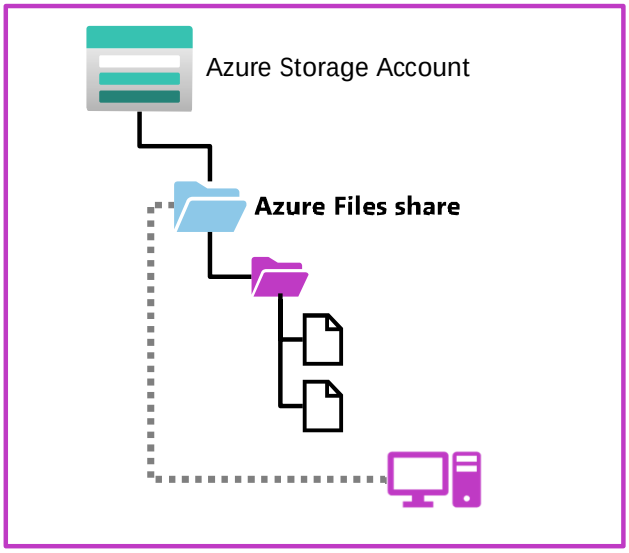
Systems like Azure Databricks can mount a distributed file system hosted in Azure Data Lake Store Gen 2 and use it to process huge volumes of data.

To create an Azure Data Lake Store Gen 2 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. Note that upgrading is a one-way process – after upgrading a storage account to support a hierarchical namespace for blob storage, you cannot revert it to a flat namespace.

Azure Files

Files shares in the cloud that can be accessed from anywhere with an internet connection

- Support for common file sharing protocols:
  - Server Message Block (SMB)
  - Network File System (NFS) – *requires premium tier*
- Data is replicated for redundancy and encrypted at rest



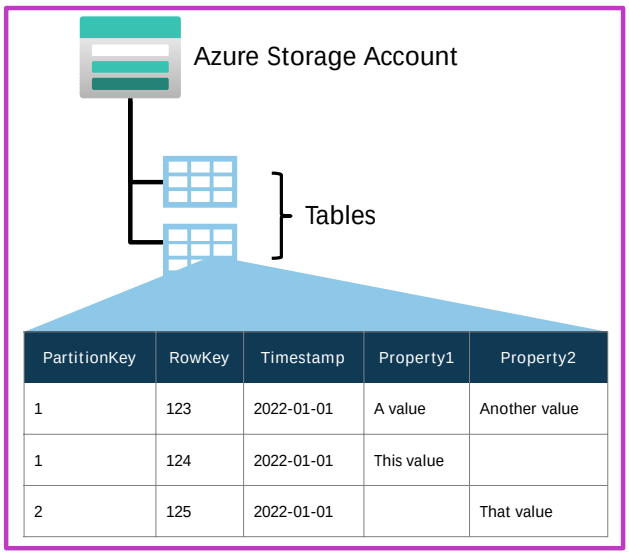
Azure Files is essentially a way to create cloud-based network shares, such as you typically find in on-premises organizations to make documents and other files available to multiple users. By hosting file shares in Azure, organizations can eliminate hardware costs and maintenance overhead, and benefit from high availability and scalable cloud storage for files.

SMB file sharing is commonly used across multiple operating systems (Windows, Linux, MacOS). NFS shares are used by Linux and MacOS versions. To create an NFS share, you must us a *premium* tier storage account and create and configure a virtual network through which access to the share can be controlled.

Azure Table Storage

Key-Value storage for application data

- Tables consist of *key* and *value* columns
  - Partition and row keys
  - Custom property columns for data values
    - A *Timestamp* column is added automatically to log data changes
- Rows are grouped into partitions to improve performance
- Property columns are assigned a data type, and can contain any value of that type
- Rows do not need to include the same property columns



PartitionKey	RowKey	Timestamp	Property1	Property2
1	123	2022-01-01	A value	Another value
1	124	2022-01-01	This value	
2	125	2022-01-01		That value

In an Azure Table Storage table, items are referred to as *rows*, and fields are known as *columns*. However, don't let this terminology confuse you by thinking that an Azure Table Storage table is like a table in a relational database. An Azure table enables you to store *semi-structured* data. All rows in a table must have a unique key (composed of a partition key and a row key), but apart from that the columns in each row can vary. Unlike traditional relational databases, Azure Table Storage tables have no concept of relationships, stored procedures, secondary indexes, or foreign keys. Data will usually be denormalized, with each row holding the entire data for a logical entity. For example, a table holding customer information might store the first name, last name, one or more telephone numbers, and one or more addresses for each customer. The number of fields in each row can be different, depending on the number of telephone numbers and addresses for each customer, and the details recorded for each address. In a relational database, this information would be split across multiple rows in several tables.

To help ensure fast access, Azure Table Storage splits a table into partitions. Partitioning is a mechanism for grouping related rows, based on a common property or *partition key*. Rows that share the same partition key will be stored together. Partitioning not only helps to organize data, it can also improve scalability and performance:

- Partitions are independent from each other, and can grow or shrink as rows are added to, or removed from, a partition. A table can contain any number of partitions.
- When you search for data, you can include the partition key in the search criteria. This helps to narrow down the volume of data to be examined, and improves performance by reducing the amount of I/O (reads and writes) needed to locate the data.

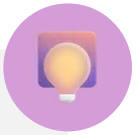
The key in an Azure Table Storage table comprises two elements; the partition key that identifies the partition containing the row (as described above), and a row key that is unique to each row in the same partition. Items in the same partition are stored in row key order. If an application adds a new row to a table, Azure ensures that the row is placed in the correct position in the table. This scheme enables an application to quickly perform *Point queries* that identify a single row, and *Range queries* that fetch a contiguous block of rows in a partition.

Lab: Explore Azure Storage

In this lab, you will provision and use Azure Storage

1. Start the virtual machine for this lab
   or go to the exercise page
   at https://go.microsoft.com/fwlink/?linkid=2261876
2. Follow the instructions to complete the exercise on Microsoft Learn
   Use the Azure subscription provided for this lab

1: Knowledge check



1. What are the elements of an Azure Table storage key?
   - ☐ Table name and column name
   - ☑ Partition key and row key
   - ☐ Row number

2. What should you do to an existing Microsoft Fabric tenant in order to support a data lake?
   - ☐ Add an Azure Files share
   - ☐ Create Azure Storage tables for the data you want to analyze
   - ☑ No action is required

3. Why might you use Azure File storage?
   - ☐ To take advantage of compatibility with major analytics systems of Azure Data Lake Store
   - ☑ To enable users at different sites to share files
   - ☐ To store large binary data files containing images or other unstructured data

Allow students a few minutes to think about the questions, and then use the animated slide to reveal the correct answers.