**Agenda: Azure Storage Service**

* About Storage Service and Account
* Creating a Storage Account
* Working with Blob Storage
  + Types of Blobs (Block, Append, Page)
  + Container and Metadata
  + Soft Copy
  + Azure Storage Explorer
  + Transfer Data using AzCopy
  + Programming Blob Storage
  + Async Blob Copy
  + Import and Export Service
  + Implement and Configure CDN
  + Configure Custom Domain
  + Scale Blob Storage
  + Storage Service Encryption
* Manage Access / Securing Storage
  + Create and Manage Shared Access Signature
  + Account SAS vs Service SAS
  + Using Stored Access Policies
  + Regenerating Keys
  + Encrypt Keys using Azure Key Vault integration
  + Programming in C#
* Working with Table Storage
  + Understanding NoSQL Database
  + Creating Table and Entities using Storage Explorer
  + Entities and Properties
  + Table storage vs COSMOS DB Table API
  + Programming Table Storage
* Azure Queues Storage
  + Understanding Async Communication using Queues
  + Comparing Azure Queues and Service Bus Queues
  + Programming Queues
* Azure SMB File Storage
  + Common usage of File Storage
  + Shares, Directory and File
  + Managing Using Azure Portal
  + Programming File Storage
* Azure File Sync
* Configure diagnostics, monitoring and analytics
  + Capturing Metrics Data
  + Analyzing Diagnostic Data
  + Capturing Logs
  + Retention Policies and Logging Levels
  + Analyze Logs
  + Enabling Monitoring
  + Enabling Alerts

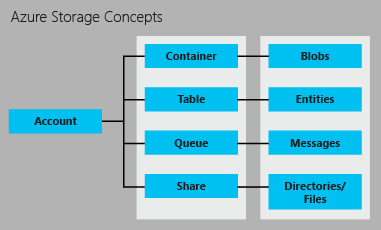
**About Azure Storage Service and Account**

Cloud computing enables new scenarios for applications requiring **scalable, durable and highly** **available** storage for their data – which is exactly why Microsoft developed **Azure Storage Service.**

* Azure Storage is a **PaaS service** that you can use to store both **unstructured** and **partially structured** data.
* **Azure Storage is massively scalable and elastic:** It can store and process **hundreds of terabytes of data** to support the big data scenarios required by scientific, financial analysis, and media applications. Or you can store the **small amounts of data** required for a small business website.
* By default, you can create up to **100 storage accounts** in a single Azure subscription. Each standard storage account can contain up to **500 TB** of combined blob, queue, table and file data.
* As the demands on your storage application grow, Azure Storage **automatically allocates** the appropriate resources to meet them. **We are charged only for what we use**.

It offers **four types of storage services**, depending on the type of data that they are designed to store:

1. **Blob Storage** stores file data. A blob can be any type of **text or binary data**, such as a document, media file, or application installer. Blob Storage is sometimes referred to as **Object storage**.
2. **File Storage** Similar to blobs, these provide storage for unstructured files, but they offer support for file sharing in the same manner as traditional on-premises Windows file shares.
3. **Table Storage** stores partially structured datasets. Table storage is a **NoSQL** key-attribute data store, which allows for rapid development and fast access to large quantities of data.
4. **Queue Storage** provides **reliable messaging** for workflow processing and for communication between components of cloud services.



**Creating Storage Account**

1. Azure Portal 🡪 Browse Storage Accounts 🡪 **New** 🡪 **Data + Storage** 🡪 **Storage account**

## Azure Storage Account

An Azure storage account is a **secure account** that gives you access to services in Azure Storage. Your storage account provides the unique namespace for your storage resources. There are two types of storage accounts:

1. A **standard storage** account includes Blob, Table, Queue, and File storage.

Standard storage accounts are backed by magnetic drives (HDD) and provide the lowest cost per GB. They are best for applications that require bulk storage or where data is accessed infrequently.

1. A **premium storage** account is ideally supposed to be used for Azure Virtual Machine disks.

Premium storage accounts are backed by solid state drives (SSD) and offer consistent low-latency performance. They can only be used with Azure virtual machine disks and are best for I/O-intensive applications, like databases. Additionally, virtual machines that use Premium storage for all disks qualify for a 99.9% SLA, even when running outside an availability set.

1. **Account Kind**: Storage ~~(general purpose v1)~~ / **StorageV2 (general purpose v2)** / ~~Blob Storage~~

* **General-purpose v2 accounts**: Basic storage account type for blobs, files, queues, and tables. Recommended for most scenarios using Azure Storage.
* **Block blob storage accounts**: Blob-only storage accounts with premium performance characteristics. Recommended for scenarios with high transactions rates, using smaller objects, or requiring consistently low storage latency.
* **FileStorage storage accounts**: Files-only storage accounts with premium performance characteristics. Recommended for enterprise or high performance scale applications.

|  |  |  |  |
| --- | --- | --- | --- |
| **Storage account type** | **Supported services** | **Supported performance tiers** | **Replication options** |
| **General-purpose V2** | Blob, File, Queue, Table, and Disk (VHD) | Standard | LRS, GRS, RA-GRS, ZRS, ZGRS RA-ZGRS |
| **Block blob storage** | Blob (block blobs and append blobs only) | Premium | LRS |
| **FileStorage** | Files only | Premium | LRS |
| **Page Blobs** | Disk (VHD) | Premium | LRS​ |

1. Performance: **Standard / Premium**

Standard use HDD Drives and Premiun use SSD Drives

Premium is used for disks of VMs (Page Blobs)

Note that it is not possible to convert a Standard storage account to Premium storage account or vice versa.

1. Access tier: Cool / **Hot**

Account kind: **Blob storage,** Performance**: Standard**

* Access tier: **Hot**, if objects will be **more** frequently accessed. This allows you to store data at a **lower access cost. Higher Storage Cost**
* Access tier: **Cool**, if objects will be **less** frequently accessed. This allows you to store data at **a lower data storage cost. Higher RW Cost**
* Acces tier: **Archive.** The archive tier is optimized for data that can tolerate **several hours of retrieval latency** and will remain in the Archive tier for at least 180 days. The archive tier is the most cost-effective option for storing data, but accessing that data is more expensive than accessing data in the hot or cool tiers. It is available at level of an individual blob only, not at the storage account level. Only block blobs and append blobs can be archived.

**Rehydrate an archived blob to an online tier:**

* To read data in archive storage, you must first change the tier of the blob to hot or cool. This process is known as rehydration and can take hours to complete.
* There are currently two rehydrate priorities, High and Standard, which can be set via the optional **x-ms-rehydrate-priority** property on a **Set Blob Tier or Copy Blob** operation.
  + **Standard priority**: The rehydration request will be processed in the order it was received and **may take up to 15 hours**.
  + **High priority**: The rehydration request will be prioritized over Standard requests and may finish **in under 1 hour.**

1. Replication:

|  |
| --- |
| **Locally redundant storage (LRS):**   * **Replicates 3 times within a single data center in a single region where Storage Account is created.** * Locally redundant storage (LRS) provides at least 99.999999999% (11 nines) durability of objects over a given year. * The replicas are spread across UDs and FDs within one storage scale unit (A storage scale unit is a collection of racks of storage nodes.) * A request returns successfully only once it has been written to all three replicas. * This architecture ensures your data is available if a hardware failure affects a single rack or when nodes are upgraded during a service upgrade. * LRS is less expensive than GRS and also offers higher throughput. * For Premium Storage accounts - This is the only option available.   **Zone-redundant storage (ZRS)**   * Replicates your data across three (3) storage clusters in a single region. Each storage cluster is physically separated from the others and resides in its own availability zone. Each availability zone, and the ZRS cluster within it, is autonomous, with separate utilities and networking capabilities. * ZRS offers durability for storage resources of at least 99.9999999999% (12 9's) over a given year. * ZRS is not yet available in all regions. * Once you have created your storage account and selected ZRS, you **cannot convert** it to use to any other type of replication, or vice versa. * Consider ZRS for scenarios that require strong consistency, strong durability, and high availability even if an outage or natural disaster renders a zonal data center unavailable. * **What happens when a zone becomes unavailable?** Your data is still accessible for both read and write operations even if a zone becomes unavailable. Microsoft recommends that you continue to follow practices for **transient fault handling**. These practices include implementing retry policies with exponential back-off. * The Archive tier for Blob Storage isn't currently supported for ZRS accounts. Unmanaged disks don't support ZRS or GZRS.   **Geo-redundant storage (GRS)**   * + GRS maintains **6 copies** of your data. 3 replicas in **primary region** and also replicates your data 3 additional times to a **secondary region** that is hundreds of miles away from the primary region.   + 99.99999999999999 (16 9s) Availablility is supported when calculated around a year.   + Data is durable even in the case of a complete regional outage or a disaster in which the primary region is not recoverable.   + If failure occurs in the primary region, Azure Storage **automatically** **failover** to the secondary region.   + An update is first committed to the primary region, where it is replicated three times. Then the update is replicated to the secondary region, where it is also replicated three times.   + Requests to write data are replicated **asynchronously** to the secondary region. It is important to note that opting for GRS does not impact latency of requests made against the primary region.   + The secondary region is **automatically determined** based on the primary region, and cannot be changed.   **Read-access geo-redundant storage (RA-GRS)**   * As with GRS, your data replicates asynchronously across two regions and synchronously within each region, yielding six copies of a storage account. * This is default option when we create a storage account. * In the event that data becomes unavailable in the primary region, your application can **read data** from the secondary region. * If your primary endpoint for the Blob service is **myaccount.blob.core.windows.net**, then your secondary endpoint is **myaccount-secondary.blob.core.windows.net**. The **access** **keys** for your storage account are the **same** for both the primary and secondary endpoints.   More Details:<https://azure.microsoft.com/en-in/documentation/articles/storage-redundancy/> |

1. Secure transfer required = **Disabled** / Enabled

If Enabled only HTTPS requests will be accepted.

This option doesn’t work with Custom Domain Names for Storage account.

1. Select Subscription, Resource Group, Location
2. Virtual network: Disabled
3. Click on "Create".

**Pricing and Billing**

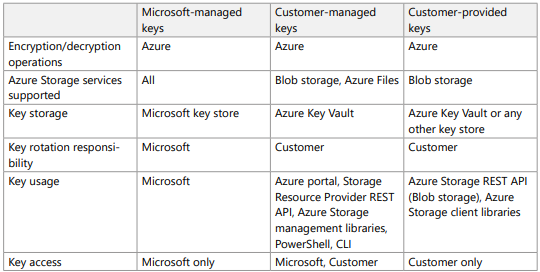
All storage accounts use a pricing model for blob storage based on the tier of each blob.

When using a storage account, the following billing considerations apply:

* **Storage costs**: In addition to, the amount of data stored, the cost of storing data varies depending on the storage tier. The per-gigabyte cost decreases as the tier gets cooler.
* **Data access costs**: Data access charges increase as the tier gets cooler. For data in the cool and archive storage tier, you are charged a per-gigabyte data access charge for reads.
* **Transaction costs**: There is a per-transaction charge for all tiers that increases as the tier gets cooler.
* **Geo-Replication data transfer costs**: This charge only applies to accounts with geo-replication configured, including GRS and RA-GRS. Geo-replication data transfer incurs a per-gigabyte charge.
* **Outbound data transfer costs**: Outbound data transfers (data that is transferred out of an Azure region) incur billing for bandwidth usage on a per-gigabyte basis, consistent with general-purpose storage accounts.
* **Changing the storage tier**: Changing the account storage tier from cool to hot incurs a charge equal to reading all the data existing in the storage account. However, changing the account storage tier from hot to cool incurs a charge equal to writing all the data into the cool tier (GPv2 accounts only).

**Azure Storage encryption for data at rest**

* Azure Storage automatically encrypts your data when persisting it to the cloud.
* Storage accounts are encrypted regardless of their performance tier (standard or premium)
* All Azure Storage resources are encrypted, including blobs, disks, files, queues, and tables. All object metadata is also encrypted.
* Encryption does not affect Azure Storage performance.
* You can rely on **Microsoft-managed keys** for the encryption of your storage account, or you can manage encryption with your own keys.
* If you choose to manage encryption with your own keys, you have two options:
  + You can specify a **customer-managed key**. It is used to encrypt all data in all services.
  + You can specify a **customer-provided key** **on Blob** storage operations. A client making a read or write request against Blob storage can include an encryption key on the request for granular control over how blob data is encrypted and decrypted.



**Working with Blob Storage**

* **Blobs** are binary large objects. The Blob service stores text and binary data.
* Blob storage is also referred to as **object storage**.
* You can use Blob storage to store content such as:
  + Documents
  + Social data such as photos, videos, music, and blogs
  + Backups of files, computers, databases, and devices
  + Images and text for web applications
  + Configuration data for cloud applications
  + Big data, such as logs and other large datasets
* Every blob is organized into a **container**. Containers also provide a useful way to assign security policies to groups of objects. A storage account can contain any number of containers, and a container can contain any number of blobs, up to the **500 TB capacity** limit of the storage account.
* **Creating BLOB Hierarchies:** The blob service in Azure Storage is based on a **flat storage scheme**. This means that creating a container one level below the **root is the only true level of container**. However, you can specify a delimiter as part of the blob name to create your own **virtual hierarchy**. For example, you could create a blob named **/January/Reports.txt** and **/February/Reports.txt**, and filter based on **/January or /February** in most tools that support Azure Storage. Most third-party storage tools allow you to create folders within a container, but they are actually being clever with the name of the blob itself.

**Blobs are addressable using the following URL format:**

http(s)://<storage account name>.**blob**.core.windows.net/<container>/<blob name>

**Types of blobs:**

1. **Block blobs** are optimized for streaming (**sequential access**) and for uploads and downloads, and are a good choice for storing documents, media files, backups etc. Azure divides data into smaller blocks of up to (64KB to) 100 megabytes (MB) in size, which subsequently **upload or download in parallel**. Individual block blobs (file) can be up to 100 GB in size. One blob can have max of 50,000 blocks.
2. **Append blobs:** Append blobs are similar to block blobs, but are optimized for **append** operations. This works best with **logging and auditing** activities. Updating or deleting of existing blocks is not supported. Max block size can be 4 MB only.
3. **Page blobs** are optimized for **random read/write** operations and provide the ability to write to a range of bytes in a blob. Blobs are accessed as **pages**, each of which is up to **512 bytes** in size. Each Page blob can be up to **8TB** each. Is best suited for **virtual machine disks (VHD)**.

**Additional Topics**

* Snapshot
* Versions
* What is Lease
* Metadata
* Soft Delete

**Index vs Metadata**

* Only index tags are automatically indexed and made searchable by the native Blob Storage service. Metadata can't be natively indexed or searched. You must use a separate service such as [Azure Search](https://docs.microsoft.com/en-us/azure/search/search-blob-ai-integration).
* Blob index tags have additional permissions for reading, filtering, and writing that are separate from the underlying blob data.
* Life cycle management filters can use index.

**Azure Blob storage lifecycle**

The lifecycle management policy lets you:

* Use **Locally Reduntant Storage (LRS) for replication**. Any other option does not support the archive tier for block blobs at this moment.
* Transition blobs to a cooler storage tier (hot to cool, hot to archive, or cool to archive) to optimize for performance and cost
* Archive Tier is applicable only for Block blobs and not for Append Blobs.
* Delete blobs at the end of their lifecycles
* Define rules to be run once per day at the storage account level.
* Apply rules to containers or a subset of blobs (using prefixes as filters)

Azure Portal 🡪 Select Storage Account 🡪 Blob Service 🡪 Lifecycle Management

**Code View:**

{

"rules": [

{

"name": "ruleFoo",

"enabled": true,

"type": "Lifecycle",

"definition": {

"filters": {

"blobTypes": [ "blockBlob" ],

"prefixMatch": [ "container1/foo" ]

},

"actions": {

"baseBlob": {

"tierToCool": { "daysAfterModificationGreaterThan": 30 },

"tierToArchive": { "daysAfterModificationGreaterThan": 90 },

"delete": { "daysAfterModificationGreaterThan": 2555 }

},

"snapshot": {

"delete": { "daysAfterCreationGreaterThan": 90 }

}

}

}

}

]

}

**Azure Storage Explorer**

Microsoft Azure Storage Explorer is a standalone app from Microsoft that allows you to easily work with Azure Storage data.

Some of the benefits of Azure Storage Explorer are:

1. Access multiple accounts and subscriptions across Azure
2. Create, delete, view, and edit storage resources.
3. View and edit Blob, Queue, Table, File, Cosmos DB storage and Data Lake Storage.
4. Obtain shared access signature (SAS) keys.
5. Available for Windows, Mac, and Linux

Azure Storage Explorer has many uses when it comes to managing your storage. See the following articles to learn more. Also, check out the videos that follow this topic.

* **Connect to an Azure subscription:** Manage storage resources that belong to your Azure subscription.
* **Work with local development storage:** Manage local storage by using the Azure Storage Emulator.
* **Attach to external storage:** Manage storage resources that belong to another Azure subscription or that are under national Azure clouds by using the storage account's name, key, and endpoints.
* **Attach a storage account by using an SAS:** Manage storage resources that belong to another Azure subscription by using a shared access signature (SAS).
* **Attach a service by using an SAS:** Manage a specific storage service (blob container, queue, or table) that belongs to another Azure subscription by using an SAS.
* **Connect to an Azure Cosmos DB account by using a connection string:** Manage Cosmos DB account by using a connection string.

**Shared Access Policy and Shared Access Signature (SAS) Token**

There are two techniques for controlling access to objects within an Azure Storage account.

1. Using the **access/authentication key and storage account name** is one technique – Gives FULL ACCESS
2. Granting access using a **shared access signature (SAS Token)** to allow granular access with expiration is another technique.

A shared access signature (SAS) is a URI that grants **restricted access rights** to Azure Storage resources. You can provide a shared access signature to clients who should not be trusted with your storage account key but whom you wish to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified **period of time**.

* A shared access signature (SAS) is a **token** that can be **appended to a URL** that enables delegated access to a storage resource.
* Anyone who possesses the token can access the resource it points to with the permissions it specifies, for the period of time that it is valid.

Azure Storage supports two kinds of shared access signatures:

1. An **Account SAS** delegates access to resources in **one or more** of the storage services. You can also delegate access to read, write, and delete operations on blob containers, tables, queues, and file shares that are not permitted with a service SAS.
2. The **Service SAS** delegates access to a resource in **just one** of the storage services: Blob, Queue, Table, or File service.

Note that **Stored Access Policies** are currently not supported for an **Account-Level SAS**.

Creating an **Account SAS** (for many operations)

1. Azure Portal 🡪 Storage Accounts 🡪 Select Account
2. Under Security and Networking Section 🡪 **Shared access signature**
3. Provide the options as required 🡪 Generate SAS
4. Copy the SAS token and share it with the client.

<https://dssdemostorage.blob.core.windows.net/container1/Azure%20Storage%20Service.pdf?sv=2017-11-09&ss=b&srt=co&sp=r&se=2022-08-13T10:05:57Z&st=2018-08-11T02:05:57Z&spr=https&sig=hFYQeZ2fvj52%2BQI0kg%2BbPmErr8J%2F5hKrGkAnF7Q7u%2F4%3D>

#### **Stored Access Policies**

A Shared Access Signature can take one of two forms:

1. **An ad-hoc SAS**. When you create an ad hoc SAS, the start time, expiration time, and permissions for the SAS are all specified on the SAS URI (or implied in the case where the start time is omitted). This type of SAS can be created on a container, blob, table, or queue.
2. **An SAS with a Stored Access Policy**. A stored access policy is defined on a resource container—a blob container, table, or queue—and can be used to manage constraints for one or more Shared Access Signatures. When you associate an SAS with a stored access policy, the SAS inherits the constraints—the start time, expiration time, and permissions—defined for the stored access policy.

**Note:** Stored access policies give you the option to **revoke permissions without having to regenerate the storage account access keys**. Set the expiration on these to be a very long time (or infinite), and make sure that it is regularly updated to move it further into the future.

**Create Ad-hoc Service SAS using Portal**

1. Azure Portal 🡪 Storage Accounts 🡪 Select Account
2. BLOB 🡪 Container 🡪 <Select Blob Item> 🡪 **Click on Generate SAS Tab**
3. Provide the details 🡪 … 🡪 Generate SAS
4. Copy the Token and use with Blob URL.

**Create Stored Access Policy using Portal**

1. Azure Portal 🡪 Storage Accounts 🡪 Select Account
2. BLOB 🡪 Container 🡪 Select the container 🡪 right click 🡪 Access Policy 🡪 + Add Policy

**Create SAS using Stored Access Policy using Portal**

1. Azure Portal 🡪 Storage Accounts 🡪 Storage Explorer (Use **Switch back to Storage Explorer** if Storage Service is visible option)

OR Use **Desktop** version of Storage Explorer:

1. BLOB 🡪 Container 🡪 Select the container / Blob 🡪 Right Click 🡪 Get Shared Access Signature
2. Select the Stored Access Policy and create the SAS.

**Summary:**

**SAS Token**

**Account SAS**

Always Ad-hoc

Same SAS token can be used for **all** services.

**Service SAS**

Ad-hoc / Stored Access Policy based

Specific to **only one** service at a time.

Container SAS token works for all blobs in that container

Blob SAS token is only for the blob for which it is generated

**Overview of Azure Active Directory for Blobs and Queues**

* When a security principal attempts to access a blob or queue resource, the request must be authorized.
* The authentication step requires that an application request an OAuth 2.0 access token at runtime.
* The authorization step requires that one or more RBAC roles be assigned to the security principal.

**Built-in RBAC roles for blobs and queues**

* Storage Blob Data Owner
* **Storage Blob Data Contributor (Read, Write, Delete and List)**
* Storage Blob Data Reader
* Storage Queue Data Contributor
* Storage Queue Data Reader
* Storage Queue Data Message Processor
* Storage Queue Data Message Sender

**Resource Scope**

The levels at which you can scope access to Azure blob and queue resources:

* The subscription
* The storage account
* The resource group
* An individual container
* An individual queue

**Azure Table Storage**

**Category**(PKCategoryId, CategoryName, ...) - SQL Table

1, Furniture

2, Pets

3, Electronics

4, Computers

…more…

**Product**(PKProductId, PKCategoryId, ProductName, Price, Quantity) - SQL Table

1, 2, Dog01, 100, 3

2, 2, Cat01, 150, 2

3, 1, Chair01, 100, 100

4, 1, Table01, 50, 20

5, 3, Laptop, 100000, 2

**ProductAttributes** (NoSQL Table) – NO FIXED SCHEMA

**PartitionKey**: 2 (Pets), **RowKey**:1 (Dog01), Age:3, Breed:Pomerian, Color:White

**PartitionKey**: 2 (Pets), **RowKey**:2 (Cat01), Age:1, Breed:Indian, Color:Black

**PartitionKey**: 1 (Furniture), **RowKey**:3 (Chair01), Weight: 10KG, Color: Brown, Type: Chair

**PartitionKey**: 4 (Electronics), **RowKey**:5 (Laptop), Weight: 30KG, Color: Black, Brand: Lenovo

**PartitionKey**: 5 (Computers), **RowKey**:5 (Laptop), Weight: 30KG, Color: Black, Brand: Lenovo

* The Azure Table storage service stores large amounts of **partially** **structured** data offering high availability and massively scalable storage.
* The service is a **NoSQL datastore** which accepts **authenticated calls** from inside and outside the Azure cloud.
* For today's Internet-based applications, NoSQL databases like Table storage offer a popular alternative to traditional relational databases.

**Common uses of the Table service include:**

* You can use Table storage to store **flexible datasets**, such as user data for web applications, address books, device information, and any other type of metadata that your service requires.
* Storing datasets that **don't** require complex joins, foreign keys, or stored procedures and can be de-normalized for fast access.
* Quickly querying data using a clustered index (Combination of PartitionKey and RowKey).
* Accessing data using the **OData protocol**.

You can use the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

* **Table**: A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. The number of tables that a storage account can contain is limited only by the storage account capacity limit.
* **Entity**: An entity is a set of properties, similar to a database row. An entity can be up to **1MB in size**.
* **Properties**: A property is a name-value pair. Each entity can include up to **252 custom properties** to store up to **1 MB** of data. Each entity also has **3 system** properties that specify a **partition key** (string upto 1KB in size)**, a row key** (string upto 1KB in size) **and a timestamp**. **Entities with the same partition key can be queried more quickly**, and inserted/updated in atomic operations. One batch operation cannot have two entities with different partition key. An entity's row key is its unique identifier within a partition.

**The URI for a specific table access is structured as follows:**

http://<account>.**table**.core.windows.net/<TableName>

**Note about Batch Operations**

* A **batch operation** is a collection of table operations which are executed by the Storage Service REST API as **a single atomic** operation.
* A batch operation may contain up to **100 individual table operations**, with the requirement that each operation entity must have **same partition key.**
* The total payload of a batch operation is limited to 4MB.
* A batch with a retrieve operation cannot contain any other operations.

**Using Visual Studio Server Explorer**

1. Server Explorer 🡪 … 🡪Select Windows Azure Storage 🡪 Select and Expand Storage Account.
2. Expand to Tables 🡪 Create Table…, Enter Name of Table
3. Select Table 🡪 Right click 🡪 View Table
4. Use the Editor to manage Table.

### Choosing Table storage or Cosmos DB Table API

Azure Cosmos DB Table API and Azure Table storage share the same table data model and expose the same create, delete, update, and query operations through their SDKs.

If you currently use Azure Table Storage, you gain the following benefits by moving to the Azure Cosmos DB Table API:

|  |  |  |
| --- | --- | --- |
|  | **Azure Table storage** | **Azure Cosmos DB Table API** |
| Latency | Fast, but no upper bounds on latency. | Single-digit millisecond latency for reads and writes, backed with <10-ms latency reads and <15-ms latency writes at the 99th percentile, at any scale, anywhere in the world. |
| Throughput | Variable throughput model. Tables have a scalability limit of 20,000 operations/s. | Highly scalable with dedicated reserved throughput per table that's backed by SLAs. Accounts have no upper limit on throughput and support >10 million operations/s per table. |
| Global distribution | Single region with one optional readable secondary read region for high availability. You can't initiate failover. | Turnkey global distribution from one to 30+ regions. Support for automatic and manual failovers at any time, anywhere in the world. |
| Indexing | Only primary index on PartitionKey and RowKey. No secondary indexes. | Automatic and complete indexing on all properties, no index management. |
| Query | Query execution uses index for primary key, and scans otherwise. | Queries can take advantage of automatic indexing on properties for fast query times. |
| Consistency | Strong within primary region. Eventual within secondary region. | Five well-defined consistency levels to trade off availability, latency, throughput, and consistency based on your application needs. |
| Pricing | Storage-optimized. | Throughput-optimized. |
| SLAs | 99.99% availability. | 99.99% availability SLA for all single region accounts and all multi-region accounts with relaxed consistency, and 99.999% read availability on all multi-region database accounts Industry-leading comprehensive SLAs on general availability. |

**Azure Queues Storage**

Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS.

A single queue message can be up to 64 KB in size, and a queue can contain millions of messages, up to the total capacity limit of a storage account (i.e. 500TB)

Queues can have metadata associated with them. Metadata is in the form of <name, value> pairs.

Common uses of Queue storage include:

1. Creating a backlog of work to process asynchronously.
2. Passing messages from an Azure App Service to an Azure Function.

* **Queue:** A queue contains a set of messages. All messages must be in a queue.
* **Message:** A message, **in any format**, of up to 64KB.

**Difference between Storage Queues and Service Bus Queue**

1. Azure storage queue does not provide publish/subscribe mechanism.
2. Azure storage queue can store data maximum for 7 days.
3. Azure storage queue does not provide a guaranteed FIFO delivery.
4. Azure storage queue does not provide transactional behavior.

**Important Notes about Storage Queues:**

* When a message is retrieved from the queue, the response includes the message and a pop receipt value, which is required to delete the message.
* The message is **not automatically deleted** from the queue, but after it has been retrieved, it is **not visible to other clients** for the time interval specified by the **visibilitytimeout** parameter. Hence every message is delivered **at-least once**.
* Every time it's received and not deleted, its **Dequeuecount is incremented**.

**Azure File Storage**

* Azure File storage is a service that offers file shares in the cloud using the standard [Server Message Block (SMB) Protocol](https://msdn.microsoft.com/library/windows/desktop/aa365233.aspx).
* With Azure File storage, you can migrate **legacy applications** that rely on file shares to Azure quickly and without costly rewrites.
* Microsoft Azure virtual machines can share file data across application components via mounted shares, and on-premises applications can access file data in a share via the File storage API.

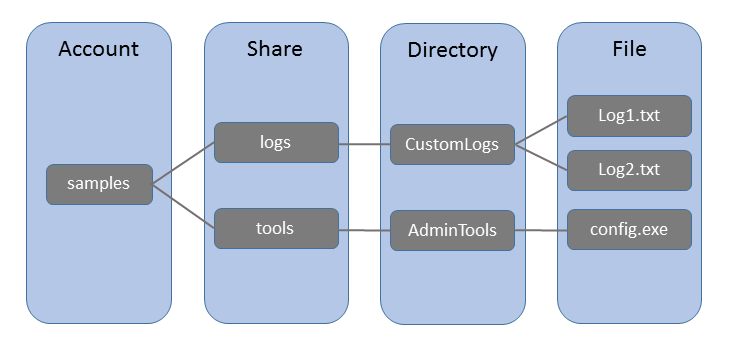
**Why Azure File is useful**

* **Replace or supplement on-premises file servers.**
* **"Lift and shift" of Legacy applications.**
* **Simplify cloud development**
  + **Shared application settings**
  + **Diagnostic share**
  + **Dev/Test/Debug**

**Common uses of File storage include:**

* Migrating on-premises applications that rely on file shares to run on Azure virtual machines or cloud services, without expensive rewrites.
* Storing shared application settings, for example in configuration files.
* Storing diagnostic data such as logs, metrics, and crash dumps in a shared location.
* Storing tools and utilities needed for developing or administering Azure virtual machines or cloud services.

## File storage concepts



* **Storage Account:** All access to Azure Storage is done through a storage account.
* **Share:** A File storage share is an SMB file share in Azure. All directories and files must be created in a parent share. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the 5 TB total capacity of the file share.
* **Directory:** An optional hierarchy of directories.
* **File:** A file in the share. A file may be up to 1 TB in size.
* **Max size of a File Share = 5TB**

**URL format:** https**://<storage account>**.file.core.windows.net/**<share>**/**<directory/directory>**/**<file>**

The following example URL could be used to address one of the files in the diagram above:  
<http://samples.file.core.windows.net/logs/CustomLogs/Log1.txt>

**Managing Using Azure Portal:**

1. Azure Portal 🡪 **Storage accounts** 🡪 Create and Choose the Storage Account

**Note: File storage is replicated only via LRS or GRS right now…**

1. Choose "Files" service 🡪 Click "+ File share" 🡪 New file share = Images, Quota=100GB
2. Optionally add directory and in the directory and upload the file[s].

**Mapping the file share in Windows**

1. Go to File Share Blade and click on **Connect** 🡪 Copy the net command edit the values
2. Open Command Prompt in Administrator Mode 🡪 Execute the net command
3. You can manage the File Share using the local drive.

**Note: Ensure port 445 is open: Azure Files uses SMB protocol. SMB communicates over TCP port 445**

**Creating File Share using Powershell**

Retrieve storage account and storage account key

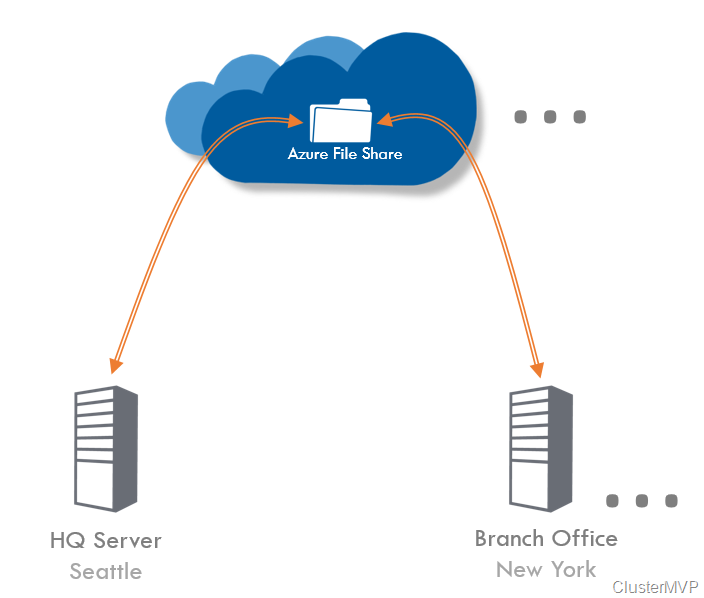
$storageContext = **New-AzureStorageContext** <storage-account-name> <storage-account-key>

# Create the file share, in this case "logs"

$share = **New-AzureStorageShare** logs -Context $storageContext

**Azure File Sync**

* Use Azure File Sync to centralize your organization's file shares in Azure Files, while keeping the flexibility, performance, and compatibility of an on-premises file server.
* Azure File Sync transforms Windows Server into a quick cache of your Azure file share.
* You can use any protocol that's available on Windows Server to access your data locally, including SMB, NFS, and FTPS.
* You can have as many caches as you need across the world.

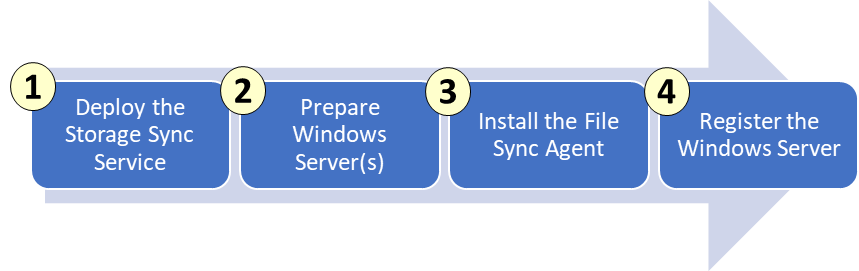


**There are many uses and advantages to file sync.**

* **Lift and shift.** The ability to move applications that require access between Azure and on-premises systems. Provide write access to the same data across Windows Servers and Azure Files. This lets companies with multiple offices have a need to share files with all offices.
* **Branch Offices**. Branch offices need to backup files, or you need to setup a new server that will connect to Azure storage.
* **Backup and Disaster Recovery**. Once File Sync is implemented, Azure Backup will back up your on-premises data. Also, you can restore file metadata immediately and recall data as needed for rapid disaster recovery.
* **File Archiving**. Only recently accessed data is located on local servers. Non-used data moves to Azure in what is called **Cloud Tiering**. Cloud Tiering files will have greyed icons with an offline 'O' file attribute to let the user know the file is only in Azure.

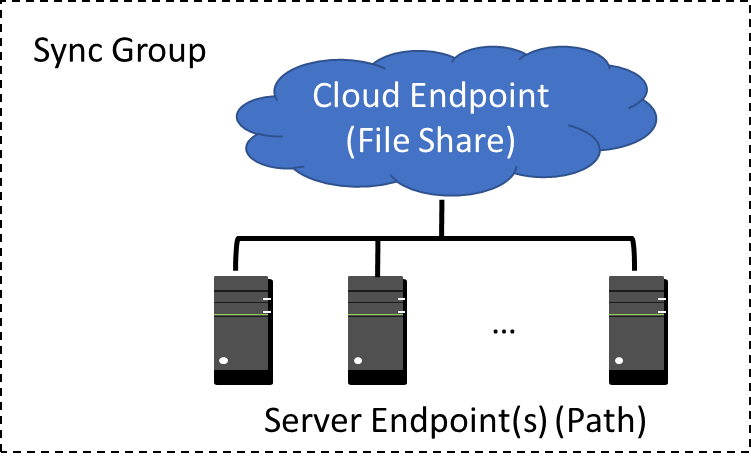
**File Sync Service Deployment**

There are a few things that need to be configured before you synchronize your files.



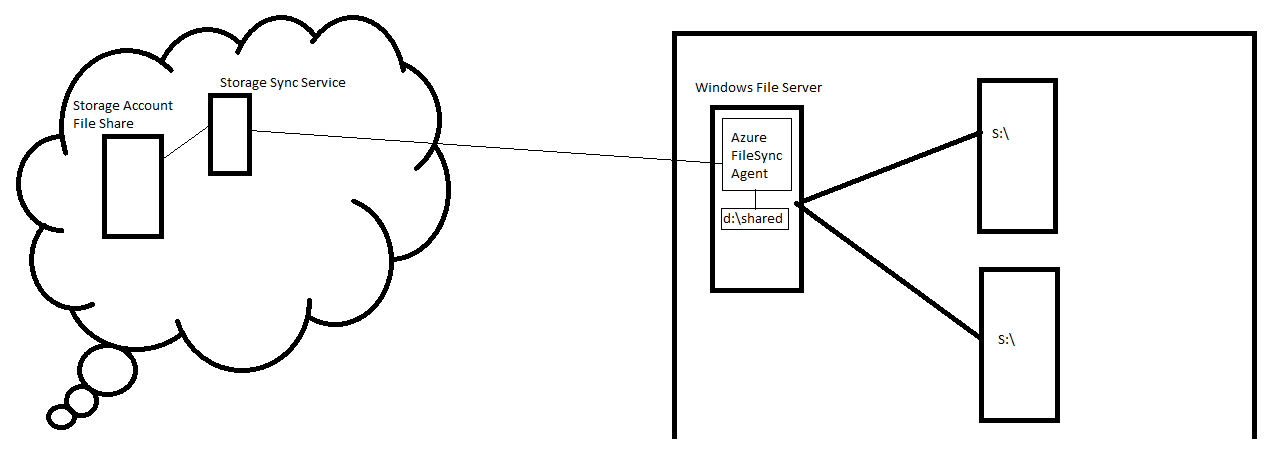
1. **Deploy the Storage Sync Service**. The Storage Sync Service is the top-level Azure resource for Azure File Sync. A distinct top-level resource from the storage account resource is required because the Storage Sync Service can create sync relationships with multiple storage accounts via multiple sync groups. A subscription can have multiple Storage Sync Service resources deployed.
2. **Prepare Windows Server to use with Azure File Sync**. For each server that you intend to use with Azure File Sync, including server nodes in a Failover Cluster, you will need to configure the server. Preparation steps include temporarily disabling Internet Explorer Enhanced Security and ensuring you have latest PowerShell version.
3. **Install the Azure File Sync Agent**. The Azure File Sync agent is a downloadable package that enables Windows Server to be synced with an Azure file share. The Azure File Sync agent installation package should install relatively quickly. We recommend that you keep the default installation path and that you enable Microsoft Update to keep Azure File Sync up to date.
4. **Register Windows Server with Storage Sync Service**. When the Azure File Sync agent installation is finished, the Server Registration UI automatically opens. Registering Windows Server with a Storage Sync Service establishes a trust relationship between your server (or cluster) and the Storage Sync Service. Registration requires your Subscription ID, Resource Group, and Storage Sync Service (created in step one). A server (or cluster) can be registered with only one Storage Sync Service at a time.

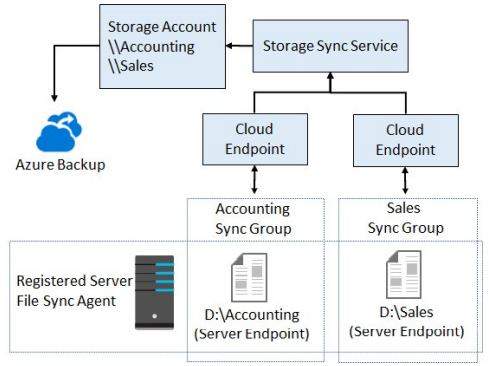
### File Sync Service Deployment (Synchronization)



A **Sync Group** defines the sync topology for a set of files. Endpoints within a sync group are kept in sync with each other. A sync group must contain at least one cloud endpoint, which represents an Azure file share created in your storage account, and at least one server endpoint, which represents a path on a Windows Server.

✔️ As Azure File Sync is available in only few Regions: Eg: West US, remember that your storage account must be located in one of the regions in which Azure File Sync is supported.

****



**Walkthrough**

1. Create Storage Account
   1. Create **File Share** in Storage Account.
2. Create **Azure File Sync** Service
   1. Azure Portal 🡪 Create a Resource 🡪 **Azure File Sync**
   2. In File Sync Service 🡪 Create Sync Group
      1. Select Storage Account and File Share created in Step1
3. Register our **On-Premise Server** or local server
   1. Create a **Windows** VM 2019 Server
   2. RDP to VM, Ensure that Azure PowerShell Cmdlets are installed.

Install-Module AzureRM

Import-Module AzureRM

* 1. Go to Server Manager 🡪 Local Server 🡪 **Turn Off IE Enanhanced Security**
  2. Open Browser 🡪 Search **Azure File Sync Agent Download** 🡪 Download and Install File Sync Agent
  3. Complete the **Registration** process: **SignIn to Azure Account** 🡪 Select the File Sync Service 🡪 Register

1. Azure Portal 🡪 File Sync Group 🡪 Sync Group 🡪 Add server endpoint 🡪 Path = **C:\AzureFiles.**

* **Registered server**. The name of the server or cluster where you want to create the server endpoint.
* **Path**. The Windows Server path to be synced as part of the sync group. The path should not be the root volume.
* **Cloud Tiering**. A switch to enable or disable cloud tiering.
* **Volume Free Space**. The amount of free space to reserve on the volume on which the server endpoint is located. For example, if volume free space is set to 50% on a volume that has a single server endpoint, roughly half the amount of data is tiered to Azure Files.

1. RDP to VM and Note that the folder is created in that VM.
2. Dump some files in that folder and note in Portal that files are now present in Azure file share.

**Implement Azure storage account failover**

<https://docs.microsoft.com/en-us/azure/storage/common/storage-initiate-account-failover>

<https://docs.microsoft.com/en-us/azure/storage/common/last-sync-time-get>