**4.1 - Configure storage accounts**

Azure Storage support three categories of data: structured data, unstructured data, and virtual machine data.

**Virtual machine data**

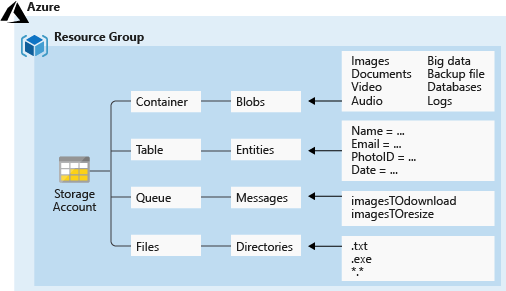
* Storage includes disks and files. Disks are persistent block storage for Azure IaaS virtual machines. Files are fully managed file shares in the cloud.
* Storage for virtual machine data is provided through Azure managed disks. Data disks are used by virtual machines to store data like database files, website static content, or custom application code. The number of data disks you can add depends on the virtual machine size.

**Unstructured data**

* This is the least organized. The format of unstructured data is referred to as nonrelational.
* Unstructured data can be stored by using Azure Blob Storage and Azure Data Lake Storage. Blob Storage is a highly scalable, REST-based cloud object store. Azure Data Lake Storage is the Hadoop Distributed File System (HDFS) as a service.

**Structured data**

* This is stored in a relational format that has a shared schema. Structured data is often contained in a database table with rows, columns, and keys. Tables are an autoscaling NoSQL store.
* Structured data can be stored by using Azure Table Storage, Azure Cosmos DB, and Azure SQL Database. Azure Cosmos DB is a globally distributed database service. Azure SQL Database is a fully managed database-as-a-service built on SQL.



**Things to consider when using Azure Storage**

* **Consider durability and availability**. Azure Storage is durable and highly available. Redundancy ensures your data is safe during transient hardware failures. You replicate data across datacenters or geographical regions for protection from local catastrophe or natural disaster. Replicated data remains highly available during an unexpected outage.
* **Consider secure access**. Azure Storage encrypts all data. Azure Storage provides you with fine-grained control over who has access to your data.
* **Consider scalability**. Azure Storage is designed to be massively scalable to meet the data storage and performance needs of modern applications.
* **Consider manageability**. Microsoft Azure handles hardware maintenance, updates, and critical issues for you.
* **Consider data accessibility**. Data in Azure Storage is accessible from anywhere in the world over HTTP or HTTPS. Microsoft provides SDKs for Azure Storage in various languages. You can use .NET, Java, Node.js, Python, PHP, Ruby, Go, and the REST API. Azure Storage supports scripting in Azure PowerShell or the Azure CLI. The Azure portal and Azure Storage Explorer offer easy visual solutions for working with your data.

**Azure Blob Storage**

Azure Blob Storage is Microsoft's object storage solution for the cloud. Blob Storage is optimized for storing massive amounts of unstructured or nonrelational data, such as text or binary data. Blob Storage is ideal for:

* Serving images or documents directly to a browser.
* Storing files for distributed access.
* Streaming video and audio.
* Storing data for backup and restore, disaster recovery, and archiving.
* Storing data for analysis by an on-premises or Azure-hosted service.

You can access data from Azure Blob Storage by using the NFS protocol.

**Azure Queue Storage**

Azure Queue Storage is used to store and retrieve messages. Queue messages can be up to 64 KB in size, and a queue can contain millions of messages. Queues are used to store lists of messages to be processed asynchronously.

**Azure Files**

Azure Files enables you to set up highly available network file shares. Shares can be accessed by using the Server Message Block (SMB) protocol and the Network File System (NFS) protocol. Multiple virtual machines can share the same files with both read and write access. You can also read the files by using the REST interface or the storage client libraries.

File shares can be used for many common scenarios:

* Many on-premises applications use file shares. This feature makes it easier to migrate those applications that share data to Azure. If you mount the file share to the same drive letter that the on-premises application uses, the part of your application that accesses the file share should work with minimal, if any, changes.
* Configuration files can be stored on a file share and accessed from multiple virtual machines. Tools and utilities used by multiple developers in a group can be stored on a file share, ensuring that everybody can find them, and that they use the same version.
* Diagnostic logs, metrics, and crash dumps are just three examples of data that can be written to a file share and processed or analyzed later.

**Azure Table Storage**

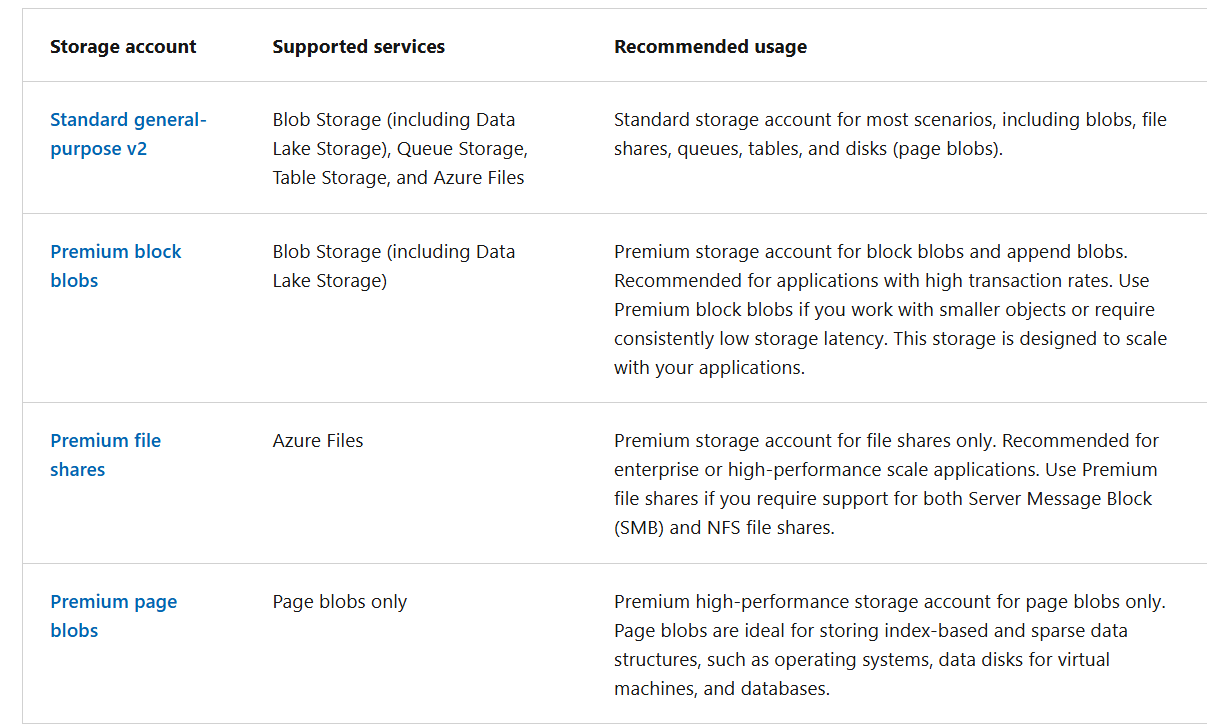
Azure Table storage is a service that stores nonrelational structured data (also known as structured NoSQL data) in the cloud, providing a key/attribute store with a schemaless design. Because Table storage is schemaless, it's easy to adapt your data as the needs of your application evolve. Access to Table storage data is fast and cost-effective for many types of applications, and is typically lower in cost than traditional SQL for similar volumes of data. In addition to the existing Azure Table Storage service, there's a new Azure Cosmos DB Table API offering that provides throughput-optimized tables, global distribution, and automatic secondary indexes.

**Things to know about storage account types**

**Standard storage accounts** are backed by magnetic hard disk drives (HDD). A standard storage account provides the lowest cost per GB. You can use Standard storage for applications that require bulk storage or where data is infrequently accessed.

**Premium storage accounts** are backed by solid-state drives (SSD) and offer consistent low-latency performance. You can use Premium storage for Azure virtual machine disks with I/O-intensive applications like databases.

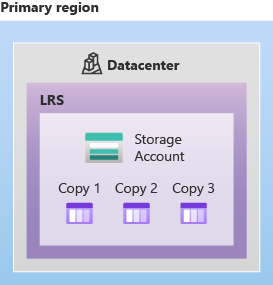
You **can't convert a Standard storage account to a Premium storage account or vice versa**. You must create a new storage account with the desired type and copy data, if applicable, to a new storage account. All storage account types are encrypted by using Storage Service Encryption (SSE) for data at rest.



**Locally redundant storage**

Locally redundant storage is the lowest-cost replication option and offers the least durability compared to other strategies. If a data center-level disaster occurs, such as fire or flooding, all replicas might be lost or unrecoverable. Despite its limitations, LRS can be appropriate in several scenarios:

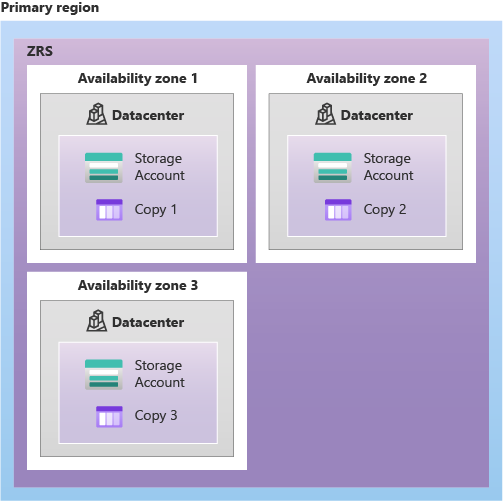
* Your application stores data that can be easily reconstructed if data loss occurs.
* Your data is constantly changing like in a live feed, and storing the data isn't essential.
* Your application is restricted to replicating data only within a location due to data governance requirements.



**Zone redundant storage**

Zone redundant storage synchronously replicates your data across three 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, and has separate utilities and networking capabilities. Storing your data in a ZRS account ensures you can access and manage your data if a zone becomes unavailable. ZRS provides excellent performance and low latency.

* ZRS isn't currently available in all regions.
* Changing to ZRS from another data replication option requires the physical data movement from a single storage stamp to multiple stamps within a region.



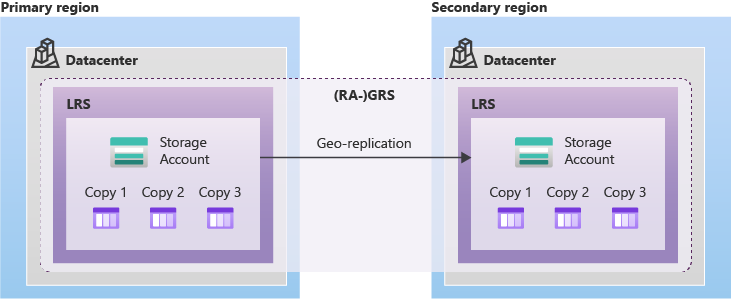
**Geo-redundant storage**

Geo-redundant storage replicates your data to a secondary region (hundreds of miles away from the primary location of the source data). GRS provides a higher level of durability even during a regional outage. GRS is designed to provide at least 99.99999999999999% (16 9's) durability. When your storage account has GRS enabled, your data is durable even when there's a complete regional outage or a disaster where the primary region isn't recoverable.

If you implement GRS, you have two related options to choose from:

* GRS replicates your data to another data center in a secondary region. The data is available to be read only if Microsoft initiates a failover from the primary to secondary region.
* Read-access geo-redundant storage (RA-GRS) is based on GRS. RA-GRS replicates your data to another data center in a secondary region, and also provides you with the option to read from the secondary region. With RA-GRS, you can read from the secondary region regardless of whether Microsoft initiates a failover from the primary to the secondary.

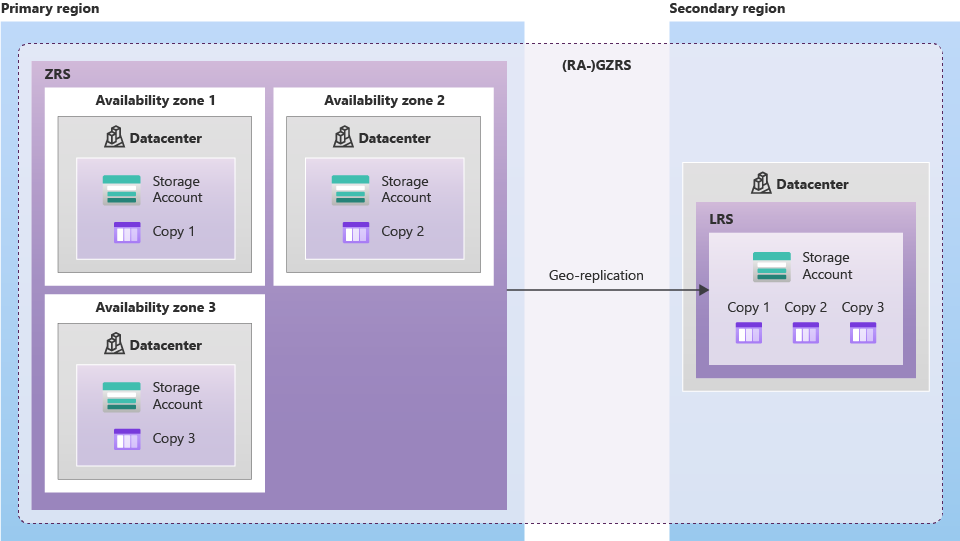
For a storage account with GRS or RA-GRS enabled, all data is first replicated with locally redundant storage. An update is first committed to the primary location and replicated by using LRS. The update is then replicated asynchronously to the secondary region by using GRS. Data in the secondary region uses LRS. Both the primary and secondary regions manage replicas across separate fault domains and upgrade domains within a storage scale unit. The storage scale unit is the basic replication unit within the datacenter. Replication at this level is provided by LRS.



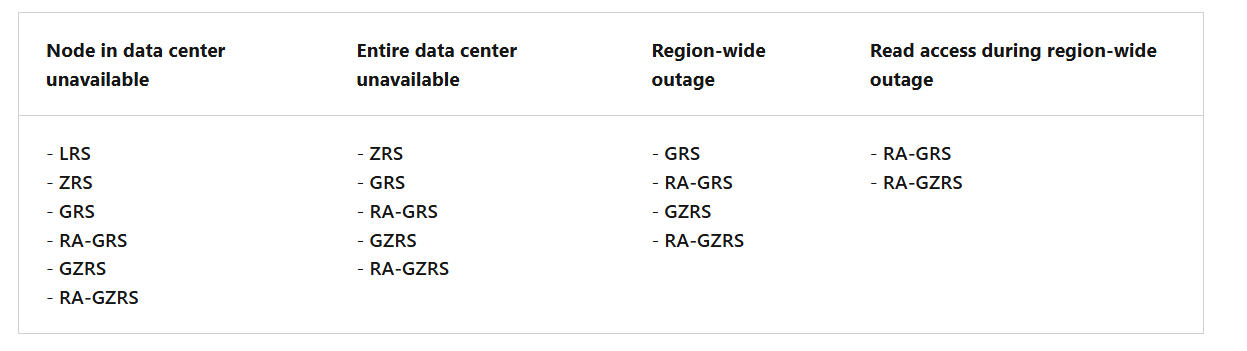
**Geo-zone redundant storage**

Geo-zone-redundant storage combines the high availability of zone-redundant storage with protection from regional outages as provided by geo-redundant storage. Data in a GZRS storage account is replicated across three Azure availability zones in the primary region, and also replicated to a secondary geographic region for protection from regional disasters. Each Azure region is paired with another region within the same geography, together making a regional pair.

With a GZRS storage account, you can continue to read and write data if an availability zone becomes unavailable or is unrecoverable. Additionally, your data is also durable during a complete regional outage or during a disaster in which the primary region isn't recoverable. GZRS is designed to provide at least 99.99999999999999% (16 9's) durability of objects over a given year. GZRS also offers the same scalability targets as LRS, ZRS, GRS, or RA-GRS. You can optionally enable read access to data in the secondary region with read-access geo-zone-redundant storage (RA-GZRS).



**Things to consider when choosing replication strategies**



**Access storage**

Every object you store in Azure Storage has a unique URL address. Your storage account name forms the subdomain portion of the URL address. The combination of the subdomain and the domain name, which is specific to each service, forms an endpoint for your storage account.



**Configure custom domains**

You can configure a custom domain to access blob data in your Azure storage account. As we reviewed, the default endpoint for Azure Blob Storage is \<storage-account-name>.blob.core.windows.net. If you map a custom domain and subdomain, such as www.contoso.com, to the blob or web endpoint for your storage account, your users can use that domain to access blob data in your storage account.

**Direct mapping** lets you enable a custom domain for a subdomain to an Azure storage account. For this approach, you create a CNAME record that points from the subdomain to the Azure storage account.

The following example shows how a subdomain is mapped to an Azure storage account to create a CNAME record in the domain name system (DNS):

* Subdomain: blobs.contoso.com
* Azure storage account: \<storage account>\.blob.core.windows.net
* Direct CNAME record: contosoblobs.blob.core.windows.net

**Secure storage endpoints**

In the Azure portal, each Azure service requires certain steps to configure the service endpoints and restrict network access.

To access these settings for your storage account, you use the Firewalls and virtual networks settings. You add the virtual networks that should have access to the service for the account. - This setting restricts access to your storage account from specific subnets on virtual networks or public IPs.

The service endpoints for a storage account provide the base URL for any blob, queue, table, or file object in Azure Storage. Use this base URL to construct the address for any given resource.

**Things to know about configuring service endpoints**

Here are some points to consider about configuring service access settings:

* You can configure the service to allow access to one or more public IP ranges.
* Subnets and virtual networks must exist in the same Azure region or region pair as your storage account.

**4.2 - Configure Azure Blob Storage**

Azure Blob Storage is a service that stores unstructured data in the cloud as objects or blobs. Blob stands for Binary Large Object. Blob Storage is also referred to as object storage or container storage.

**Things to know about Azure Blob Storage**

Blob Storage can store any type of text or binary data. Some examples are text documents, images, video files, and application installers.

Blob Storage uses three resources to store and manage your data:

* An Azure storage account
* Containers in an Azure storage account
* Blobs in a container

To implement Blob Storage, you configure several settings:

* Blob container options.
* Blob types and upload options.
* Blob Storage access tiers.
* Blob lifecycle rules.
* Blob object replication options.

**Things to consider when implementing Azure Blob Storage**

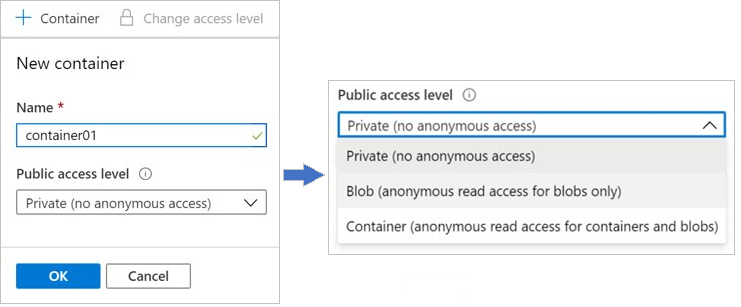
* Consider browser uploads. Use Blob Storage to serve images or documents directly to a browser.
* Consider distributed access. Blob Storage can store files for distributed access, such as during an installation process.
* Consider streaming data. Stream video and audio by using Blob Storage.
* Consider archiving and recovery. Blob Storage is a great solution for storing data for backup and restore, disaster recovery, and archiving.
* Consider application access. You can store data in Blob Storage for analysis by an on-premises or Azure-hosted service.

**Configuration characteristics of containers and blobs.**

* All blobs must be in a container.
* Containers organize your blob storage.
* A container can store an unlimited number of blobs.
* An Azure storage account can contain an unlimited number of containers.
* You must create a storage container before you can begin to upload data.

**Configure a container**

In the Azure portal, you configure settings to create a container for an Azure storage account. As you review these details, consider how you might organize containers in your storage account.



Name: Enter a name for your container. The name must be unique within the Azure storage account.

* The name can contain only lowercase letters, numbers, and hyphens.
* The name must begin with a letter or a number.
* The minimum length for the name is three characters.
* The maximum length for the name is 63 characters.

Public access level: The access level specifies whether the container and its blobs can be accessed publicly. By default, container data is private and visible only to the account owner. There are three access level choices:

* Private: (Default) Prohibit anonymous access to the container and blobs.
* Blob: Allow anonymous public read access for the blobs only.
* Container: Allow anonymous public read and list access to the entire container, including the blobs.

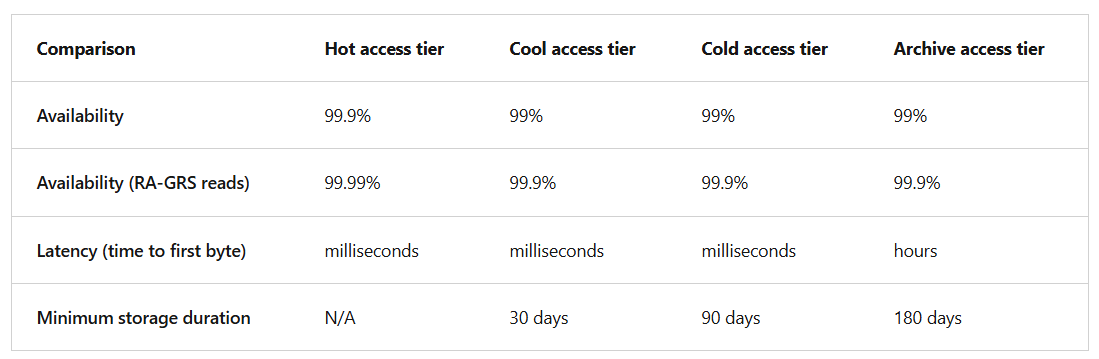
**Assign blob access tiers**

Azure Storage supports several access tiers for blob data. These tiers include Hot, Cool, Cold, and Archive. Each access tier is optimized to support a particular pattern of data usage.

**Things to know about blob access tiers**

* **Hot tier** is optimized for frequent reads and writes of objects in the Azure storage account. A good usage case is data that is actively being processed. An online tier optimized for storing data that is accessed or modified frequently. The hot tier has the highest storage costs, but the lowest access costs.
* **Cool tier** is optimized for storing large amounts of infrequently accessed data. This tier is intended for data that remains in the Cool tier for at least 30 days. A usage case for the Cool tier is short-term backup and disaster recovery datasets and older media content. This content shouldn't be viewed frequently, but it needs to be immediately available. Storing data in the Cool tier is more cost-effective. The cool tier has lower storage costs and higher access costs compared to the hot tier.
* **Cold tier** is also optimized for storing large amounts of infrequently accessed data. This tier is intended for data that can remain in the tier for at least 90 days. The cold tier has lower storage costs and higher access costs compared to the cool tier.
* **Archive tier** is an offline tier that's optimized for data that can tolerate several hours of retrieval latency. Data must remain in the Archive tier for at least 180 days or be subject to an early deletion charge. Data for the Archive tier includes secondary backups, original raw data, and legally required compliance information. This tier is the most cost-effective option for storing data. Accessing data is more expensive in the Archive tier than accessing data in the other tiers.

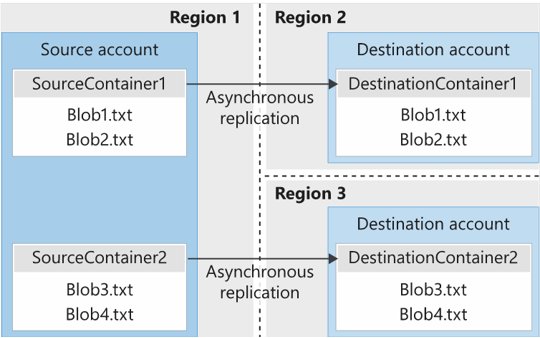
**Compare access tiers**



**Determine blob object replication**

Object replication copies blobs in a container asynchronously according to policy rules that you configure.

Replication includes the blob content, metadata properties, and versions. The following illustration shows an example of asynchronous replication of blob containers between regions.



**Things to know about blob object replication**

* Object replication requires that blob versioning is enabled on both the source and destination accounts. When blob versioning is enabled, you can access earlier versions of a blob. This access lets you recover your modified or deleted data.
* Object replication doesn't support blob snapshots. Any snapshots on a blob in the source account aren't replicated to the destination account.
* Object replication is supported when the source and destination accounts are in the Hot, Cool, or Cold tier. The source and destination accounts can be in different tiers.
* When you configure object replication, you create a replication policy that specifies the source Azure storage account and the destination storage account.
* A replication policy includes one or more rules that specify a source container and a destination container. The policy identifies the blobs in the source container to replicate.

**Things to consider when configuring blob object replication**

There are many benefits to using blob object replication. Consider the following scenarios and think about how replication can be a part of your Blob Storage strategy.

* Consider latency reductions. Minimize latency with blob object replication. You can reduce latency for read requests by enabling clients to consume data from a region that's in closer physical proximity.
* Consider efficiency for compute workloads. Improve efficiency for compute workloads by using blob object replication. With object replication, compute workloads can process the same sets of blobs in different regions.
* Consider data distribution. Optimize your configuration for data distribution. You can process or analyze data in a single location and then replicate only the results to other regions.
* Consider costs benefits. Manage your configuration and optimize your storage policies. After your data is replicated, you can reduce costs by moving the data to the Archive tier by using lifecycle management policies.

**Consider versioning when using object replication**

You can enable Blob versioning to automatically maintain previous versions of an object. When blob versioning is enabled, you can access earlier versions of a blob. This access lets you recover your modified or deleted data.

**Manage blobs**

A blob can be any type of data and any size file. Azure Storage offers three types of blobs: block blob, page blob, and append blob.

**Things to know about blob types**

* **Block blob** consists of blocks of data that are assembled to make a blob. Most Blob Storage scenarios use block blobs. **Block blobs are ideal for storing text and binary data in the cloud**, like files, images, and videos. The **block blob type is the default type for a new blob**. When you're creating a new blob, if you don't choose a specific type, the new blob is created as a block blob.
* **Append blob** is similar to a block blob because the append blob also consists of blocks of data. The blocks of data in an append blob are optimized for append operations. **Append blobs are useful for logging scenarios**, where the amount of data can increase as the logging operation continues.
* **Page blob** can be up to 8 TB in size. Page blobs are more efficient for frequent read/write operations. **Azure Virtual Machines uses page blobs for operating system disks** and data disks.

**Things to consider when managing blob storage**

You can use the portal to upload and manage blobs. This option is good for a few files. After you identify the files to upload, you choose the blob type and block size, and the container folder. You also set the access tier and the encryption scope.

For larger numbers of files, it's best to use a tool. Review the following options and consider which tools would suit your configuration needs.

* **Azure Storage Explorer**. Upload, download, and manage blobs, files, queues, and tables, as well as Azure Data Lake Storage entities and managed disks. You can also view, edit, and manage resources, preview data, and configure storage permissions and access controls.
* **AzCopy**. An easy-to-use command-line tool for Windows and Linux. You can copy data to and from Blob Storage, across containers, and across storage accounts.
* **Azure Data Box Disk**. A service for transferring on-premises data to Blob Storage when large datasets or network constraints make uploading data over the wire unrealistic. You can use Azure Data Box Disk to request solid-state disks (SSDs) from Microsoft. You can copy your data to those disks and ship them back to Microsoft to be uploaded into Blob Storage.

**Determine Blob Storage pricing**

Understanding your access patterns and correlating them with your durability and availability needs helps you to best manage your Azure Blob Storage costs. The primary tool for estimating these costs is the Azure pricing calculator. The pricing tool can calculate migration, monthly estimates, and future pricing estimates based on the workload-driven input that you specify. In general, the cost of block blob storage depends on:

* Volume of data stored per month.
* Quantity and types of operations performed, along with any data transfer costs.
* Data redundancy option selected.

**Things to know about pricing for Blob Storage**

Review the following billing considerations for an Azure storage account and Blob Storage.

* **Performance tiers**. The Blob Storage tier determines the amount of data stored and the cost for storing that data. As the performance tier gets cooler, the per-gigabyte cost decreases.
* **Data access costs**. Data access charges increase as the tier gets cooler. For data in the Cool and Archive tiers, you're billed a per-gigabyte data access charge for reads.
* **Transaction costs**. There's a per-transaction charge for all tiers. The charge increases as the tier gets cooler.
* **Geo-replication data transfer costs**. This charge only applies to accounts that have geo-replication configured, including GRS and RA-GRS. Geo-replication data transfer incurs a per-gigabyte charge.
* **Outbound data transfer costs**. Outbound data transfers incur billing for bandwidth usage on a per-gigabyte basis. This billing is consistent with general-purpose Azure storage accounts.
* **Changes to the storage tier**. If you change the account storage tier from Cool to Hot, you incur a charge equal to reading all the data existing in the storage account. 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).

**4.3 - Configure Azure Storage security**

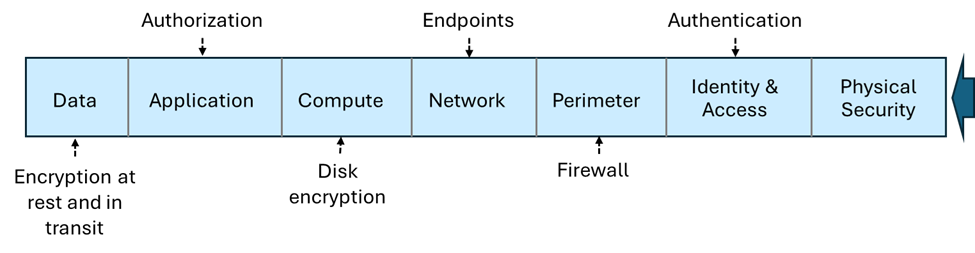
**Review Azure Storage security strategies**

Administrators use different strategies to ensure their data is secure. Common approaches include encryption, authentication, authorization, and user access control with credentials, file permissions, and private signatures. Azure Storage offers a suite of security capabilities based on common strategies to help you secure your data.

**Things to consider when using authorization security**

* Microsoft Entra ID is Microsoft's cloud-based identity and access management service. With Microsoft Entra ID, you can assign fine-grained access to users, groups, or applications by using role-based access control.
* Shared Key authorization relies on your Azure storage account access keys and other parameters to produce an encrypted signature string. The string is passed on the request in the Authorization header.
* A SAS delegates access to a particular resource in your Azure storage account with specified permissions and for a specified time interval.
* You can optionally make blob resources public at the container or blob level. A public container or blob is accessible to any user for anonymous read access. Read requests to public containers and blobs don't require authorization.

**Things to know about Azure Storage security strategies**



* **Encryption at rest**. Storage Service Encryption (SSE) with a 256-bit Advanced Encryption Standard (AES) cipher encrypts all data written to Azure Storage. When you read data from Azure Storage, Azure Storage decrypts the data before returning it. This process incurs no extra charges and doesn't degrade performance. Encryption at rest includes encrypting virtual hard disks (VHDs) with Azure Disk Encryption. This encryption uses BitLocker for Windows images, and uses dm-crypt for Linux.
* **Encryption in transit**. Keep your data secure by enabling transport-level security between Azure and the client. Always use HTTPS to secure communication over the public internet. When you call the REST APIs to access objects in storage accounts, you can enforce the use of HTTPS by requiring secure transfer for the storage account. After you enable secure transfer, connections that use HTTP will be refused. This flag will also enforce secure transfer over SMB by requiring SMB 3.0 for all file share mounts.
* **Encryption models**. Azure supports various encryption models, including server-side encryption that uses service-managed keys, customer-managed keys in Key Vault, or customer-managed keys on customer-controlled hardware. With client-side encryption, you can manage and store keys on-premises or in another secure location.
* **Authorize requests**. For optimal security, Microsoft recommends using Microsoft Entra ID with managed identities to authorize requests against blob, queue, and table data, whenever possible. Authorization with Microsoft Entra ID and managed identities provides superior security and ease of use over Shared Key authorization.
* **RBAC** ensures that resources in yourstorage account are accessible only when you want them to be, and to only thoseusers or applications whom you grant access. Assign RBAC roles scoped to anAzure storage account.
* **Storage analytics**. Azure Storage Analyticsperforms logging for a storage account. You can use this data to tracerequests, analyze usage trends, and diagnose issues with your storage account.

**Create shared access signatures**

* A shared access signature (SAS) is a uniform resource identifier (URI) that grants restricted access rights to Azure Storage resources. SAS is a secure way to share your storage resources without compromising your account keys.
* You can provide a SAS to clients who shouldn't have access to your storage account key. By distributing a SAS URI to these clients, you grant them access to a resource for a specified period of time. You'd typically use a SAS for a service where users read and write their data to your storage account.
* A user delegation SAS is secured with Microsoft Entra credentials and also by the permissions specified for the SAS. A user delegation SAS is supported for Blob Storage and Data Lake Storage,
* An account-level SAS to allow access to anything that a service-level SAS can allow, plus other resources and abilities. For example, you can use an account-level SAS to allow the ability to create file systems.
* A service-level SAS to allow access to specific resources in a storage account. You'd use this type of SAS, for example, to allow an app to retrieve a list of files in a file system, or to download a file.
* A stored access policy can provide another level of control when you use a service-level SAS on the server side. You can group SASs and provide other restrictions by using a stored access policy.

**Recommendations for managing risks**

* Always use HTTPS for creation and distribution
* Reference stored access policies where possible
* Set near-term expiry times for an unplanned SAS
* Require clients automatically renew the SAS
* Plan carefully for the SAS start time
* Define minimum access permissions for resources
* Validate data written by using a SAS
* Don't assume a SAS is always the correct choice

**Identify URI and SAS parameters**

When you create your shared access signature (SAS), a uniform resource identifier (URI) is created by using parameters and tokens. The URI consists of your Azure Storage resource URI and the SAS token.

**Things to know about URI definitions**

* https://myaccount.blob.core.windows.net/?restype=service&comp=properties&sv=2015-04-05&ss=bf&st=2015-04-29T22%3A18%3A26Z&se=2015-04-30T02%3A23%3A26Z&sr=b&sp=rw&sip=168.1.5.60-168.1.5.70&spr=https&sig=F%6GRVAZ5Cdj2Pw4tgU7IlSTkWgn7bUkkAg8P6HESXwmf%4B
* **Resource URI** - https://myaccount.blob.core.windows.net/ ?restype=service &amp;comp=properties
* **Storage version - sv**=2015-04-05
* **Storage service - ss**=bf
* **Start time - st**=2015-04-29T22%3A18%3A26Z
* **Expiry time - se**=2015-04-30T02%3A23%3A26Z
* **Resource - sr**=b
* **Permissions - sp**=rw
* **IP range - sip**=168.1.5.60-168.1.5.70
* **Protocol - spr**=https
* **Signature - sig**=F%6GRVAZ5Cdj2Pw4tgU7Il STkWgn7bUkkAg8P6HESXwmf%4B

**Determine Azure Storage encryption**

* Azure Storage encryption for data at rest protects your data by ensuring your organizational security and compliance commitments are met. The encryption and decryption processes happen automatically. Because your data is secured by default, you don't need to modify your code or applications.
* When you create a storage account, Azure generates two 512-bit storage account access keys for that account. These keys can be used to authorize access to data in your storage account via Shared Key authorization, or via SAS tokens that are signed with the shared key.
* Microsoft recommends that you use Azure Key Vault to manage your access keys, and that you regularly rotate and regenerate your keys. Using Azure Key Vault makes it easy to rotate your keys without interruption to your applications. You can also manually rotate your keys.

**Things to know about Azure Storage encryption**

* Data is automatically encrypted before writtento Azure storage.
* Data is automatically decrypted when retrieved.
* Azure Storage encryption, encryption at rest,decryption, and key management are transparent to users.
* All data written to Azure Storage is encrypted through 256-bit advanced encryption standard (AES) encryption. AES is one of the strongest block ciphers available.
* Azure Storage encryption is enabled for all new and existing storage accounts and can't be disabled.

**Configure Azure Storage encryption**

* In the Azure portal, you configure Azure Storage encryption by specifying the encryption type. You can manage the keys yourself, or you can have the keys managed by Microsoft. Consider how you might implement Azure Storage encryption for your storage security.
* **Infrastructure encryption**. Infrastructure encryption can be enabled for the entire storage account, or for an encryption scope within an account. When infrastructure encryption is enabled for a storage account or an encryption scope, data is encrypted twice—once at the service level and once at the infrastructure level—with two different encryption algorithms and two different keys.
* **Platform-managed keys**. Platform-managed keys (PMKs) are encryption keys generated, stored, and managed entirely by Azure. Customers don't interact with PMKs. The keys used for Azure Data Encryption-at-Rest, for instance, are PMKs by default.
* **Customer-managed keys**. Customer managed keys (CMK), on the other hand, are keys read, created, deleted, updated, and/or administered by one or more customers. Keys stored in a customer-owned key vault or hardware security module (HSM) are CMKs. Bring Your Own Key (BYOK) is a CMK scenario in which a customer imports (brings) keys from an outside storage location.

**Create customer-managed keys**

* For your Azure Storage security solution, you can use Azure Key Vault to manage your encryption keys. The Azure Key Vault APIs can be used to generate encryption keys. You can also create your own encryption keys and store them in a key vault.

**Things to know about customer-managed keys**

* By creating your own keys (referred to as customer-managed keys), you have more flexibility and greater control.
* You can create, disable, audit, rotate, and define access controls for your encryption keys.
* Customer-managed keys can be used with Azure Storage encryption. You can use a new key or an existing key vault and key. The Azure storage account and the key vault must be in the same region, but they can be in different subscriptions.

**Configure customer-managed keys**

* In the Azure portal, you can configure customer-managed encryption keys. You can create your own keys, or you can have the keys managed by Microsoft. Consider how you might use Azure Key Vault to create your own customer-managed encryption keys.
* Encryption type: Choose how the encryption key is managed: by Microsoft or by yourself (customer).
* Encryption key: Specify an encryption key by entering a URI, or select a key from an existing key vault.

**Apply Azure Storage security best practices**

* Storage insights provides comprehensive monitoring of your Azure Storage accounts. Storage Insights delivers a unified view of your Azure Storage services performance, capacity, and availability.

**What are the benefits of Storage insights?**

* **Detailed Metrics and Logs**. Azure Storage Insights offers detailed metrics, logs, and diagnostic information that enhance visibility into storage operations. This helps in monitoring key performance indicators (KPIs) such as latency, throughput, capacity utilization, and transactions.
* **Enhanced Security and Compliance**. By using Azure Storage Insights, you can ensure enhanced security and compliance. It provides actionable insights and alerts that help in swiftly identifying and resolving security issues.
* **Role-Based Access Control (RBAC)**. Azure Storage Insights integrates with Azure's security features, including role-based access control (RBAC), Microsoft Entra ID, connection strings, and access control list (ACL) permissions. RBAC ensures secure access to your data and resources.
* **Unified View**. It delivers a unified view of your Azure Storage services' performance, capacity, and availability, which is crucial for maintaining the security and efficiency of your storage accounts.

**Security uses for Storage Insights**

* **Real-Time Monitoring**. Azure Storage Insights enables real-time monitoring of storage accounts, allowing you to track usage trends, monitor performance, and set up alerts for any anomalies.
* **Security Auditing**. It aids in security auditing by providing comprehensive monitoring and detailed logs, which are essential for ensuring compliance and identifying any security issues.
* **Health Analysis and Optimization**. The tool helps in health analysis and optimization of storage accounts, ensuring security and optimal performance.

**4.4 - Configure Azure Files**

**Compare storage for file shares and blob data**

* Azure Files offers fully managed file shares in the cloud. You can access Azure file shares by using the Server Message Block (SMB), Network File System (NFS), and HTTP protocols. Clients can connect to Azure file shares from Windows, Linux, and macOS devices.

**Things to know about Azure Files**

* **Serverless deployment**. An Azure file share is a PaaS offering of a fully managed file share that doesn't require any infrastructure. You don't need to take care of any VMs, operating systems, or updates.
* **Almost unlimited storage**. A single Azure file share can store up to 100 tebibytes (TiB) of files, and a file can be up to 4 TiB in size. The files are organized in a hierarchical folder structure in the same way as on on-premises file servers.
* **Data encryption**. The data on an Azure file share is encrypted at rest in an Azure datacenter and in transit on a network.
* **Access from anywhere**. By default, clients can access Azure file shares from anywhere if they have internet connectivity.
* **Integration into an existing environment**. You can control access to Azure file shares by using Microsoft Entra identities or AD DS identities that are synced to Microsoft Entra ID. This helps ensure that users can have the same experience accessing an Azure file share as when they access an on-premises file server.
* **Previous versions and backups**. You can create Azure file share snapshots that integrate with the Previous Versions feature in File Explorer. You can also use Azure Backup to back up Azure file shares.
* **Data redundancy**. Azure file share data replicates to multiple locations in the same Azure datacenter or across many Azure datacenters. The replication setting of the Azure storage account that includes the file share controls the data redundancy.

**Things to consider when using Azure Files**

* **Consider replacement and supplement options**. Replace or supplement traditional on-premises file servers or NAS devices by using Azure Files.
* **Consider global access**. Directly access Azure file shares by using most operating systems, such as Windows, macOS, and Linux, from anywhere in the world.
* **Consider lift and shift support**. Lift and shift applications to the cloud with Azure Files for apps that expect a file share to store file application or user data.
* **Consider using Azure File Sync**. Replicate Azure file shares to Windows Servers by using Azure File Sync. You can replicate on-premises or in the cloud for performance and distributed caching of the data where it's being used.
* **Consider shared applications**. Store shared application settings such as configuration files in Azure Files.
* **Consider diagnostic data**. Use Azure Files to store diagnostic data such as logs, metrics, and crash dumps in a shared location.
* **Consider tools and utilities**. Azure Files is a good option for storing tools and utilities that are needed for developing or administering Azure VMs or cloud services.

**Manage Azure file shares**

* Azure Files offers two industry-standard file system protocols for mounting Azure file shares: the Server Message Block (SMB) protocol and the Network File System (NFS) protocol. Azure file shares don't support both the SMB and NFS protocols on the same file share, although you can create SMB and NFS Azure file shares within the same storage account.

**Types of Azure file shares**

* **Premium file shares** store data on solid-state drives (SSDs), and are available only in the FileStorage storage account kind. They provide consistent high performance and low latency, and are available in LRS redundancy, with ZRS available in some regions. Not available in all Azure regions.
* **Standard file shares** store data on hard disk drives (HDDs) and deploy in the general-purpose version 2 (GPv2) storage account type. Provide performance for workloads such as general-purpose file shares and dev/test environments. Standard file shares are available for LRS, ZRS, GRS, and GZRS, in all Azure regions.

**Types of authentication**

* **Identity-based authentication over SMB** - Provides the same seamless single sign-on (SSO) experience when accessing Azure file shares as accessing on-premises file shares.
* **Access key** - An access key is an older and less flexible option. An Azure storage account has two access keys that can be used when making a request to the storage account, including to Azure Files. Access keys are static and provide full control access to Azure Files. Access keys should be secured and not shared with users, because they bypass all access control restrictions. A best practice is to avoid sharing storage account keys and use identity-based authentication whenever possible.
* **A Shared Access Signature (SAS) token** - SAS is a dynamically generated Uniform Resource Identifier (URI) that's based on the storage access key. SAS provides restricted access rights to an Azure storage account. Restrictions include allowed permissions, start and expiry time, allowed IP addresses from where requests can be sent, and allowed protocols. With Azure Files, a SAS token is only used to provide REST API access from code.

**Creating SMB Azure file shares**

* **Open port 445**. SMB communicates over TCP port 445. Make sure port 445 is open. Also, make sure your firewall isn't blocking TCP port 445 from the client machine. If you can't unblock port 445, then a VPN or ExpressRoute connection from on-premises to your Azure network is required, with Azure Files exposed on your internal network using private endpoints.
* **Enable secure transfer**. The Secure transferrequired setting enhances the security of your storage account by limitingrequests to your storage account from secure connections only. Consider thescenario where you use REST APIs to access your storage account. If you attemptto connect, and secure transfer required is enabled, you must connect by usingHTTPS. If you try to connect to your account by using HTTP, and secure transferrequired is enabled, the connection is rejected.

**Mount an SMB Azure file share on Windows**

* You can use Azure file shares seamlessly in Windows and Windows Server. You can connect to your Azure file share with Windows or Windows Server in the Azure portal. Specify the Drive where you want to mount the share, and choose the Authentication method. The Azure portal supplies you with PowerShell commands to run when you're ready to work with the file share.

**Mount SMB Azure file share on Linux**

* You can also connect to Azure file shares from Linux machines. From your virtual machine page, select Connect. SMB Azure file shares can be mounted in Linux distributions by using the CIFS kernel client. File mounting can be done on-demand with the mount command or on-boot (persistent) by creating an entry in /etc/fstab.

**Create file share snapshots**

* Azure Files provides the capability to take share snapshots of file shares. File share snapshots capture a point-in-time, read-only copy of your data.

**Things to know about file share snapshots**

* The Azure Files share snapshot capability is provided at the file share level.
* Share snapshots are incremental in nature. Only data changed since the most recent share snapshot is saved.
* Incremental snapshots minimize the time requiredto create share snapshots and saves on storage costs.
* Even though share snapshots are saved incrementally, you only need to retain the most recent share snapshot to restore the share.
* You can retrieve a share snapshot for anindividual file. This level of support helps with restoring individual filesrather than having to restore to the entire file share.
* If you delete a file share that has share snapshots, all of its snapshots are deleted along with the share.

**Things to consider when using file share snapshots**

* Protect against application error and data corruption
* Protect against accidental deletions or unintended changes
* Support backup and recovery

**Implement soft delete for Azure Files**

* Azure Files offers soft delete for file shares. Soft delete lets you recover deleted files and file shares.

**Things to know about soft delete for Azure Files**

* Soft delete for file shares is enabled at the storage account level.
* Soft delete transitions content to a soft deleted state instead of being permanently erased.
* Soft delete lets you configure the retention period. The retention period is the amount of time that soft deleted file shares are stored and available for recovery.
* Soft delete provides a retention period between 1 and 365 days.
* Soft delete can be enabled on either new or existing file shares.

**Things to consider when using soft delete for Azure Files**

* **Recover from accidental data loss**. You can recover deleted or corrupted data with soft delete.
* **Upgrade scenarios**. Use soft delete to restore to a known good state after a failed upgrade attempt.
* **Ransomware protection**. Use soft delete to recover data without paying ransom to cybercriminals.
* **Long-term retention**. Use soft delete to comply with data retention requirements.
* **Business continuity**. Use soft delete to prepare your infrastructure to be highly available for critical workloads.

**Azure Storage Explorer**

* Azure Storage Explorer is a standalone application that makes it easy to work with Azure Storage data on Windows, macOS, and Linux. With Azure Storage Explorer, you can access multiple accounts and subscriptions, and manage all your Storage content.

**Things to know about Azure Storage Explorer**

* Azure Storage Explorer requires both management (Azure Resource Manager) and data layer permissions to allow full access to your resources. You need Microsoft Entra ID permissions to access your storage account, the containers in your account, and the data in the containers.
* Azure Storage Explorer lets you connect to different storage accounts.
* Connect to storage accounts associated with your Azure subscriptions.
* Connect to storage accounts and services that are shared from other Azure subscriptions.
* Connect to and manage local storage by using the Azure Storage Emulator.

**Consider Azure File Sync**

* Azure File Sync enables you to cache several Azure Files shares on an on-premises Windows Server or cloud virtual machine. You can 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.

**Things to know about Azure File Sync**

* Azure File Sync transforms Windows Server into a quick cache of your Azure Files shares.
* You can use any protocol that's available on Windows Server to access your data locally with Azure File Sync, including SMB, NFS, and FTPS.
* Azure File Sync supports as many caches as you need around the world.

**Cloud tiering**

* Cloud tiering is an optional feature of AzureFile Sync. Frequently accessed files are cached locally on the server while allother files are tiered to Azure Files based on policy settings.
* When a file is tiered, Azure File Sync replaces the file locally with a pointer. A pointer is commonly referred to as a reparse point. The parse point represents a URL to the file in Azure Files.
* When a user opens a tiered file, Azure File Sync seamlessly recalls the file data from Azure Files without the user needing to know that the file is stored in Azure.
* Cloud tiering files have greyed icons with an offline O file attribute to let the user know when the file is only in Azure.

**Things to consider when using Azure File Sync**

* **Consider application lift and shift**. Use Azure File Sync 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.
* **Consider support for branch offices**. Support your branch offices that need to back up files by using Azure File Sync. Use the service to set up a new server that connects to Azure storage.
* **Consider backup and disaster recovery**. After you implement Azure File Sync, Azure Backup backs up your on-premises data. Restore file metadata immediately and recall data as needed for rapid disaster recovery.
* **Consider file archiving with cloud tiering**. Azure File Sync stores only recently accessed data on local servers. Implement cloud tiering so older data moves to Azure Files.