

# Overview

## About Blob Storage

Blob Storage is designed for:

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

## Azure Data Lake Storage Gen2 ([Create Azure Data Lake Storage Gen2](#))

Blob Storage supports Azure Data Lake Storage Gen2, Microsoft's enterprise big data analytics solution for the cloud. Azure Data Lake Storage Gen2 offers a hierarchical file system as well as the advantages of Blob Storage, including:

- Low-cost, tiered storage
- High availability
- Strong consistency
- Disaster recovery capabilities

## Azure Storage data services

(Use each of these services: blob,files,queues and table)

File share: [create](#) and [mount](#)  
[Queue](#)  
[Table](#)

The Azure Storage platform includes the following data services:

**Azure Blobs:** A massively scalable object store for text and binary data. Also includes support for big data analytics through Data Lake Storage Gen2.

**Azure Files:** Managed file shares for cloud or on-premises deployments.

**Azure Queues:** A messaging store for reliable messaging between application components.

**Azure Tables:** A NoSQL store for schemaless storage of structured data.

**Azure Disks:** Block-level storage volumes for Azure VMs.

Authorization methods  
(try switching authorization methods)  
(use each of the methods)

- **Azure Active Directory (Azure AD)**
- **Shared Key**
- **Shared access signatures (SAS)**
- **Active Directory Domain Services**

Encryption

There are two basic kinds of encryption available for Azure Storage.

**Encryption at rest**

Azure Storage automatically encrypts all data prior to persisting to the storage account and decrypts it prior to retrieval.

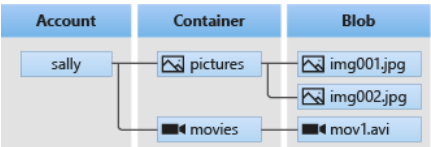
**Client-side encryption**

The Azure Storage client libraries provide methods for encrypting data from the client library before sending it across the wire and decrypting the response. Data encrypted via client-side encryption is also encrypted at rest by Azure Storage.

Introduction to Azure Blob Storage  
[\(Create storage account, container and upload a blob\)](#)  
(Check the URI for blob)

Blob Storage offers three types of resources:

- The storage account
- A container in the storage account
- A blob in a container



**Blobs**

Azure Storage supports three types of blobs:

	<ul style="list-style-type: none"><li>• <b>Block blobs store text and binary data.</b> Block blobs are made up of blocks of data that can be managed individually. Block blobs can store up to about 190.7 TiB.</li><li>• <b>Append blobs are made up of blocks like block blobs, but are optimized for append operations.</b> Append blobs are ideal for scenarios such as logging data from virtual machines.</li><li>• <b>Page blobs store random access files up to 8 TiB in size.</b> Page blobs store virtual hard drive (VHD) files and serve as disks for Azure virtual machines. For more information about page blobs, see <a href="#">Overview of Azure page blobs</a></li></ul> <p>The URI for a blob is similar to:</p> <p><code>https://myaccount.blob.core.windows.net/mycontainer/myblob</code></p>
<b>Storage account overview</b>	
Types of storage accounts ( <b>Create premium blob, file and page blobs</b> )	<ul style="list-style-type: none"><li>• Standard (general purpose v2)</li><li>• Premium (blob, file, page blobs)</li></ul>

	Type of storage account	Supported storage services	Redundancy options	Usage
	Standard general-purpose v2	Blob Storage (including Data Lake Storage <sup>1</sup> ), Queue Storage, Table Storage, and Azure Files	Locally redundant storage (LRS) / geo-redundant storage (GRS) / read-access geo-redundant storage (RA-GRS)  Zone-redundant storage (ZRS) / geo-zone-redundant storage (GZRS) / read-access geo-zone-redundant storage (RA-GZRS) <sup>2</sup>	Standard storage account type for blobs, file shares, queues, and tables. Recommended for most scenarios using Azure Storage. If you want support for network file system (NFS) in Azure Files, use the premium file shares account type.
	Premium block blobs <sup>3</sup>	Blob Storage (including Data Lake Storage <sup>1</sup> )	LRS  ZRS <sup>2</sup>	Premium storage account type for block blobs and append blobs. Recommended for scenarios with high transaction rates or that use smaller objects or require consistently low storage latency. <a href="#">Learn more about example workloads.</a>
	Premium file shares <sup>3</sup>	Azure Files	LRS  ZRS <sup>2</sup>	Premium storage account type for file shares only. Recommended for enterprise or high-performance scale applications. Use this account type if you want a storage account that supports both Server Message Block (SMB) and NFS file shares.
	Premium page blobs <sup>3</sup>	Page blobs only	LRS	Premium storage account type for page blobs only. <a href="#">Learn more about page blobs and sample use cases.</a>

	<div><div>Performance ⓘ *</div><div><div><input type="radio"/> Standard: Recommended for most scenarios (general-purpose v2 account)</div><div><input checked="" type="radio"/> Premium: Recommended for scenarios that require low latency.</div></div></div> <div><div>Premium account type ⓘ *</div><div>Redundancy ⓘ *</div></div> <div><div>Block blobs</div><div><div>Block blobs: Best for high transaction rates or low storage latency</div><div>File shares: Best for enterprise or high-performance applications that need to scale</div><div>Page blobs: Best for random read and write operations</div></div></div> <div><div>Review + create</div><div><div>&lt; Previous</div><div>Next : Advanced &gt;</div></div></div>
<div>Authorize access to data in Azure Storage</div>	

Understand authorization for data operations

The following table describes the options that Azure Storage offers for authorizing access to data:

Azure artifact	Shared Key (storage account key)	Shared access signature (SAS)	Azure Active Directory (Azure AD)	On-premises Active Directory Domain Services	Anonymous public read access	Storage Local Users
Azure Blobs	Supported	Supported	Supported	Not supported	Supported but not recommended	Supported, only for SFTP
Azure Files (SMB)	Supported	Not supported	Supported, only with AAD Domain Services	Supported, credentials must be synced to Azure AD	Not supported	Supported
Azure Files (REST)	Supported	Supported	Not supported	Not supported	Not supported	Not supported
Azure Queues	Supported	Supported	Supported	Not Supported	Not supported	Not supported
Azure Tables	Supported	Supported	Supported	Not supported	Not supported	Not supported

Authorize with Shared Key (using shared key perform some operations)

**Protect your access keys:**

Your storage account access keys are similar to a root password for your storage account. Always be careful to protect your access keys. Use Azure Key Vault to manage and rotate your keys securely. Avoid distributing access keys to other users, hard-coding them, or saving them anywhere in plain text that is accessible to others.

**Authorize with Shared Key:**

Every request made against a storage service must be authorized, unless the request is for a blob or container resource that has been made available for public or signed access. One option for authorizing a request is by using Shared Key.

Authorize access to blobs using Azure Active Directory (using ad perform some operations, give a user access to container)

Azure Storage supports using Azure Active Directory (Azure AD) to authorize requests to blob data. With Azure AD, you can use Azure role-based access control (Azure RBAC) to grant permissions to a security principal, which may be a user, group, or application service principal.

**Resource scope**

You can scope access to Azure blob resources at the following levels, beginning with the narrowest scope:


	<ul style="list-style-type: none"> <li>• <b>An individual container.</b> At this scope, a role assignment applies to all of the blobs in the container, as well as container properties and metadata.</li> <li>• <b>The storage account.</b> At this scope, a role assignment applies to all containers and their blobs.</li> <li>• <b>The resource group.</b> At this scope, a role assignment applies to all of the containers in all of the storage accounts in the resource group.</li> <li>• <b>The subscription.</b> At this scope, a role assignment applies to all of the containers in all of the storage accounts in all of the resource groups in the subscription.</li> <li>• <b>A management group.</b> At this scope, a role assignment applies to all of the containers in all of the storage accounts in all of the resource groups in all of the subscriptions in the management group.</li> </ul>
<p>Delegate access by using a shared access signature</p> <p>(define, modify and revoke stored access policy)</p> <p>Create SAS via portal</p> <p><a href="#">Create SAS with powershell</a></p> <p><a href="#">Create account SAS</a></p> <p>Create <a href="#">container/blob</a> SAS</p> <p><a href="#">User delegation SAS</a></p>	<p>A shared access signature (SAS) is a URI that grants restricted access rights to Azure Storage resources.</p> <h2>Types of shared access signatures</h2> <p>Azure Storage supports the following types of shared access signatures:</p> <ul style="list-style-type: none"> <li>• <b>An account SAS</b>, introduced with version 2015-04-05. This type of SAS delegates access to resources in one or more of the storage services.</li> <li>• <b>A service SAS</b>. This type of SAS delegates access to a resource in just one of the storage services: Azure Blob Storage, Azure Queue Storage, Azure Table Storage, or Azure Files.</li> <li>• <b>A user delegation SAS</b>, introduced with version 2018-11-09. This type of SAS is secured with Azure Active Directory credentials. It's supported for Blob Storage only.</li> </ul> <p><b>Define a stored access policy:</b></p> <p>A stored access policy provides an additional level of control over service-level shared access signatures (SASs) on the server side. Establishing a stored access policy serves to group shared access signatures and to provide additional restrictions for signatures that are bound by the policy.</p> <p><b>Create or modify a stored access policy</b></p> <p>The access policy for a shared access signature consists of the start time, expiry time, and permissions for the signature. You can specify either of the following options or combine them:</p> <ul style="list-style-type: none"> <li>• All of these parameters on the signature URI and none on the stored access policy</li> <li>• All of these parameters on the stored access policy and none on the URI</li> </ul>

**Modify or revoke a stored access policy:**  
If your existing policy grants read and write permissions to a resource, you can modify it to grant only read permissions for all future requests. In this case, the signed identifier of the new policy, as specified by the ID field, would be identical to the signed identifier of the policy that you're replacing.

To revoke a stored access policy, you can delete it, rename it by changing the signed identifier, or change the expiry time to a value in the past. Changing the signed identifier breaks the associations between any existing signatures and the stored access policy.

Add screenshots for access policy

Home > Storage accounts > mynew8978 | Containers > newcontainer

 **newcontainer** | Access policy

Container

Search

Save

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

**Access policy**

Properties

Metadata

Stored access policies

+ Add policy

Identifier	Start time
mypolicy	2/22/2023, 12:00:00 AM

Immutable blob storage ⓘ

+ Add policy

Identifier
No results

Security recommendations for Blob storage



Perform all the security recommendations

## Data protection

- Use the Azure Resource Manager deployment model
- [Enable Microsoft Defender for all of your storage accounts](#)
- [Turn on soft delete for blobs](#)
- [Turn on soft delete for containers](#)
- [Lock storage account to prevent accidental or malicious deletion or configuration changes](#)
- 
- [Store business-critical data in immutable blobs: Configure legal holds and time-based retention policies to store blob data in a WORM \(Write Once, Read Many\) state.](#)
- Require secure transfer (HTTPS) to the storage account
- Limit shared access signature (SAS) tokens to HTTPS connections only.

## Identity and access management

- Use Azure Active Directory (Azure AD) to authorize access to blob data
- Keep in mind the principle of least privilege when assigning permissions to an Azure AD security principal via Azure RBAC
- Use a user delegation SAS to grant limited access to blob data to clients
- Secure your account access keys with Azure Key Vault
- Regenerate your account keys periodically
- Disallow Shared Key authorization
- Keep in mind the principle of least privilege when assigning permissions to a SAS
- Have a revocation plan in place for any SAS that you issue to clients
- If a service SAS is not associated with a stored access policy, then set the expiry time to one hour or less
- Disable anonymous public read access to containers and blobs

## Networking

	<ul style="list-style-type: none"> <li>• Configure the minimum required version of Transport Layer Security (TLS) for a storage account.</li> <li>• Enable the Secure transfer required option on all of your storage accounts</li> <li>• Enable firewall rules</li> <li>• Allow trusted Microsoft services to access the storage account</li> <li>• Use private endpoints</li> <li>• Use VNet service tags</li> <li>• Limit network access to specific networks</li> <li>• Configure network routing preference</li> </ul> <p><b>Logging/Monitoring</b></p> <ul style="list-style-type: none"> <li>• Track how requests are authorized</li> <li>• Set up alerts in Azure Monitor</li> </ul>
Azure Storage encryption for data at rest	
About encryption key management (use both options <b>customer-managed key</b> and <b>customer-provided key</b> )	<p><b>Data in a new storage account is encrypted with Microsoft-managed keys by default.</b> You can continue to rely on Microsoft-managed keys for the encryption of your data, or you can manage encryption with your own keys.</p> <p>If you choose to manage encryption with your own keys, you have two options. You can use either type of key management, or both:</p> <ul style="list-style-type: none"> <li>• You can specify a <b>customer-managed key</b> to use for encrypting and decrypting data in Blob Storage and in Azure Files.<sup>1,2</sup> Customer-managed keys must be stored in Azure Key Vault or Azure Key Vault Managed Hardware Security Model (HSM). The managed identity that is associated with the storage account must have these permissions at a minimum to access a customer-managed key in Azure Key Vault: <i>wrapkey / unwrapkey / get</i></li> <li>• You can specify a <b>customer-provided key</b> 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. <b>Customer-provided keys</b> can be stored in Azure Key Vault or in another key store.</li> </ul>

<b>Doubly encrypt data with infrastructure encryption</b> <a href="#">(use the doubly encryption)</a>	Customers who require high levels of assurance that their data is secure can also enable 256-bit AES encryption at the Azure Storage infrastructure level. When infrastructure encryption is enabled, data in a storage account is <b>encrypted twice</b> — <b>once at the service level and once at the infrastructure level</b> — with two different encryption algorithms and two different keys. Double encryption of Azure Storage data protects against a scenario where one of the encryption algorithms or keys may be compromised. In this scenario, the additional layer of encryption continues to protect your data.
Encryption scopes for Blob storage <a href="#">(Create encryption scopes for container and a blob)</a>	<p>Encryption scopes enable you to manage encryption with a key that is scoped to a container or an individual blob.</p> <p><b>How encryption scopes work</b></p> <p><b>By default, a storage account is encrypted with a key that is scoped to the entire storage account.</b> When you define an encryption scope, you specify a key that <b>may be scoped to a container or an individual blob</b>. When the encryption scope is applied to a blob, the blob is encrypted with that key. When the encryption scope is applied to a container, it serves as the default scope for blobs in that container, so that all blobs that are uploaded to that container may be encrypted with the same key.</p> <p><b>Infrastructure encryption</b></p> <p>Infrastructure encryption in Azure Storage enables double encryption of data. With infrastructure encryption, data is encrypted twice — once at the service level and once at the infrastructure level — with two different encryption algorithms and two different keys.</p> <p>Infrastructure encryption is supported for an encryption scope, as well as at the level of the storage account. If infrastructure encryption is enabled for an account, then any encryption scope created on that account automatically uses infrastructure encryption. If infrastructure encryption is not enabled at the account level, then you have the option to enable it for an encryption scope at the time that you create the scope. The infrastructure encryption setting for an encryption scope cannot be changed after the scope is created.</p>
<b>Data protection overview</b>	

<p>Overview of data protection options</p> <p><b>(Enable/Test all of these options)</b></p> <p><b>(take and restore blob snapshots)</b></p>	<p><b>Azure Resource Manager lock:</b> Prevent a storage account from being deleted or modified.</p> <p><b>Immutability policy on a blob version or container:</b> Prevent a blob version or container from being deleted for an interval that you control.</p> <p><b>Container or blob soft delete:</b> Restore a deleted container or blob within a specified interval.</p> <p><b>Blob versioning:</b> Automatically save the state of a blob in a previous version when it's overwritten.</p> <p><b>Point-in-time restore:</b> Restore a set of block blobs to a previous point in time.</p> <p><b>Blob snapshot:</b> Manually save the state of a blob at a given point in time. A blob may be restored from a snapshot if the blob is overwritten. If the blob is deleted, snapshots are also deleted.</p>
<p>Soft delete for containers</p> <p><b>(enable soft delete for container)</b></p>	<p>Container soft delete protects your data from being accidentally deleted by maintaining the deleted data in the system for a specified period of time.</p> <p>When you enable container soft delete, you can <b>specify a retention period</b> for deleted containers that is between 1 and 365 days. The default retention period is seven days. During the retention period, you can recover a deleted container by calling the Restore Container operation.</p> <p>When you restore a container, the container's blobs and any blob versions and snapshots are also restored. However, you can only use container soft delete to restore blobs if the container itself was deleted. <b>To restore a deleted blob when its parent container hasn't been deleted, you must use blob soft delete or blob versioning.</b></p>
<p>Soft delete for blobs</p> <p><b>(enable soft delete for blobs and test overwrites with and without versioning)</b></p>	<p>Blob soft delete protects an individual blob, snapshot, or version from accidental deletes or overwrites by maintaining the deleted data in the system for a specified period of time. During the retention period, you can restore a soft-deleted object to its state at the time it was deleted.</p> <p><b>How overwrites are handled when soft delete is enabled</b></p> <p>Calling an operation such as <a href="#">Put Blob</a>, <a href="#">Put Block List</a>, or <a href="#">Copy Blob</a> overwrites the data in a blob. When blob soft delete is enabled, overwriting a blob automatically creates <b>a soft-deleted snapshot of the blob's state prior to the write operation</b>. When the retention period expires, the soft-deleted snapshot is permanently deleted.</p> <p><b>Blob soft delete and versioning</b></p> <p>If blob versioning and blob soft delete are both enabled for a storage account, then overwriting a blob automatically creates a new previous version that reflects the blob's state before the write operation. The new version isn't soft-deleted and isn't removed when the soft-delete retention period expires. No soft-deleted snapshots are created.</p>

	<p>Worth noting:</p> <p><b>Containers and blobs in the deleted account aren't recoverable.</b></p> <p><b>Blobs in the deleted container aren't recoverable.</b></p>
<p>Blob versioning (enable blob version and check how diff versions are stored)</p>	<p>You can enable Blob storage versioning to automatically maintain previous versions of an object. When blob versioning is enabled, you can access earlier versions of a blob to recover your data if it is modified or deleted.</p>
<p>Point-in-time restore for block blobs (enable point in time restore and test it)</p>	<p>Point-in-time restore provides protection against accidental deletion or corruption by enabling you to restore block blob data to an earlier state.</p> <p>Point-in-time restore is useful in scenarios where a user or application accidentally deletes data or where an application error corrupts data. Point-in-time restore also enables testing scenarios that require reverting a data set to a known state before running further tests.</p> <p><b>Prerequisites for point-in-time restore:</b></p> <p>Point-in-time restore requires that the following Azure Storage features be enabled before you can enable point-in-time restore:</p> <ul style="list-style-type: none"> <li>• <a href="#">Soft delete</a></li> <li>• <a href="#">Change feed</a></li> <li>• <a href="#">Blob versioning</a></li> </ul> <p><b>Retention period for point-in-time restore</b></p> <p>When you enable point-in-time restore for a storage account, you specify <b>a retention period</b>. Block blobs in your storage account can be restored during the retention period.</p> <p>The retention period for point-in-time restore must be at least one day less than the retention period specified for soft delete. For example, if the soft delete retention period is set to 7 days, then the point-in-time restore retention period may be between 1 and 6 days</p>

<p>Change feed support in Azure Blob Storage</p> <p><b>(enable and test change feed)</b></p>	<p>The purpose of the change feed is to provide transaction logs of all the changes that occur to the blobs and the blob metadata in your storage account. The change feed provides ordered, guaranteed, durable, immutable, read-only log of these changes.</p>
<p>Store business-critical blob data with immutable storage</p> <p><b>(apply time-based and legal hold policies)</b></p>	<p>Immutable storage for Azure Blob Storage enables users to store business-critical data in a WORM (Write Once, Read Many) state. While in a WORM state, data cannot be modified or deleted for a user-specified interval. By configuring immutability policies for blob data, you can protect your data from overwrites and deletes.</p> <p>Immutable storage for Azure Blob Storage supports two types of immutability policies:</p> <ul style="list-style-type: none"> <li>• <b>Time-based retention policies:</b> With a time-based retention policy, users can set policies to store data for a specified interval. When a time-based retention policy is set, objects can be created and read, but not modified or deleted. After the retention period has expired, objects can be deleted but not overwritten. To learn more about time-based retention policies, see <a href="#">Time-based retention policies for immutable blob data</a>.</li> <li>• <b>Legal hold policies:</b> A legal hold stores immutable data until the legal hold is explicitly cleared. When a legal hold is set, objects can be created and read, but not modified or deleted. To learn more about legal hold policies, see <a href="#">Legal holds for immutable blob data</a>.</li> </ul>
<h2>Azure Storage redundancy</h2>	
<p>(Try out read options for GRS and GZRS)</p>	<p><b>Redundancy in the primary region</b></p> <ul style="list-style-type: none"> <li>• <b>Locally redundant storage (LRS)</b> copies your data synchronously three times within a single physical location in the primary region. LRS is the least expensive replication option, but isn't recommended for applications requiring high availability or durability. LRS provides at least 99.999999999% (<b>11 nines</b>) durability of objects over a given year.</li> </ul>

	<ul style="list-style-type: none"> <li>• <b>Zone-redundant storage (ZRS)</b> copies your data synchronously across three Azure availability zones in the primary region. For applications requiring high availability, Microsoft recommends using ZRS in the primary region, and also replicating to a secondary region. ZRS offers durability for storage resources of at least 99.9999999999% <b>(12 9's)</b> over a given year.</li> </ul> <h3>Redundancy in a secondary region</h3> <p>Azure Storage offers two options for copying your data to a secondary region:</p> <ul style="list-style-type: none"> <li>• <b>Geo-redundant storage (GRS)</b> copies your data synchronously three times within a single physical location in the primary region using LRS. It then copies your data asynchronously to a single physical location in the secondary region. Within the secondary region, your data is copied synchronously three times using LRS. GRS offers durability for storage resources of at least 99.99999999999999% <b>(16 9's)</b> over a given year.</li> <li>• <b>Geo-zone-redundant storage (GZRS)</b> copies your data synchronously across three Azure availability zones in the primary region using ZRS. It then copies your data asynchronously to a single physical location in the secondary region. Within the secondary region, your data is copied synchronously three times using LRS. GZRS is designed to provide at least 99.99999999999999% <b>(16 9's)</b> durability of objects over a given year.</li> </ul> <h3>Read access to data in the secondary region</h3> <p>Geo-redundant storage (with GRS or GZRS) replicates your data to another physical location in the secondary region to protect against regional outages. With an account configured for GRS or GZRS, data in the secondary region is not directly accessible to users or applications, unless a failover occurs. The failover process updates the DNS entry provided by Azure Storage so that the secondary endpoint becomes the new primary endpoint for your storage account. During the failover process, your data is inaccessible. After the failover is complete, you can read and write data to the new primary region.</p>
<p><b>Access tiers for blob data</b> (create a blob in hot,cool and archive tier, rehydrate blob in archive tier)</p>	<ul style="list-style-type: none"> <li>• <b>Hot tier</b> - 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.</li> <li>• <b>Cool tier</b> - An online tier optimized for storing data that is infrequently accessed or modified. Data in the cool tier should be stored for a <b>minimum of 30 days</b>. The cool tier has lower storage costs and higher access costs compared to the hot tier.</li> </ul>

	<ul style="list-style-type: none"><li>● <b>Archive tier</b> - An offline tier optimized for storing data that is rarely accessed, and that has flexible latency requirements, on the order of hours. Data in the archive tier should be stored for a minimum of <b>180 days</b>. While a blob is in the archive tier, it can't be read or modified. To read or download a blob in the archive tier, you must first rehydrate it to an online tier, either hot or cool. Data in the archive tier can take up to 15 hours to rehydrate, depending on the priority you specify for the rehydration operation.</li></ul>
<p>Data lifecycle</p> <p>(create a lifecycle policy based on each points mentioned on the right)</p>	<p>With the lifecycle management policy, you can:</p> <ul style="list-style-type: none"><li>● Transition blobs from cool to hot immediately when they're accessed, to optimize for performance.</li><li>● <b>Transition current versions of a blob, previous versions of a blob, or blob snapshots to a cooler storage</b> tier if these objects haven't been accessed or modified for a period of time, to optimize for cost. In this scenario, the lifecycle management policy can move objects from hot to cool, from hot to archive, or from cool to archive.</li><li>● Delete current versions of a blob, previous versions of a blob, or blob snapshots at the end of their life cycles.</li><li>● Define rules to be run once per day at the storage account level.</li><li>● Apply rules to containers or to a subset of blobs, using name prefixes or <a href="#">blob index tags</a> as filters.</li></ul> <p><b>Lifecycle management policy definition</b></p>



	<div>JSON</div> <pre>{   "rules": [     {       "name": "rule1",       "enabled": true,       "type": "Lifecycle",       "definition": {...}     },     {       "name": "rule2",       "type": "Lifecycle",       "definition": {...}     }   ] }</pre>
<div>Object replication for block blobs</div> <div>(Create replication policies)</div>	<div>Object replication asynchronously copies block blobs between a source storage account and a destination account. Some scenarios supported by object replication include:</div> <div><ul style="list-style-type: none"><li>• <b>Minimizing latency.</b> Object replication can reduce latency for read requests by enabling clients to consume data from a region that is in closer physical proximity.</li><li>• <b>Increase efficiency for compute workloads.</b> With object replication, compute workloads can process the same sets of block blobs in different regions.</li><li>• <b>Optimizing data distribution.</b> You can process or analyze data in a single location and then replicate just the results to additional regions.</li><li>• <b>Optimizing costs.</b> After your data has been replicated, you can reduce costs by moving it to the archive tier using life cycle management policies.</li></ul></div> <div>Prerequisites and caveats for object replication</div> <div>Object replication requires that the following Azure Storage features are also enabled:</div> <div><ul style="list-style-type: none"><li>• <b>Change feed:</b> Must be enabled on the source account. To learn how to enable change feed, see <a href="#">Enable and disable the change feed</a>.</li></ul></div>

- [Blob versioning](#): Must be enabled on both the source and destination accounts. To learn how to enable versioning, see [Enable and manage blob versioning](#).

**Object replication is supported for general-purpose v2 storage accounts and premium block blob accounts.** Both the source and destination accounts must be either general-purpose v2 or premium block blob accounts. **Object replication supports block blobs only; append blobs and page blobs aren't supported.**

**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 policies and rules

**When you configure object replication, you create a replication policy that specifies the source storage account and the destination account.** A replication policy includes one or more rules that specify a source container and a destination container and indicate which block blobs in the source container will be replicated.

After you configure object replication, Azure Storage checks the change feed for the source account periodically and asynchronously replicates any write or delete operations to the destination account.

### Replication rules

Replication rules specify how Azure Storage will replicate blobs from a source container to a destination container. You can specify up to 1000 replication rules for each replication policy.

## Policy definition file

An object replication policy is defined by JSON file. You can get the policy definition file from an existing object replication policy.

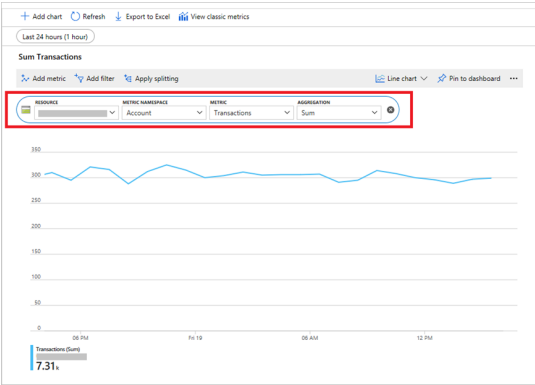
The following example defines a replication policy on the destination account with a single rule that matches the prefix *b* and sets the minimum creation time for blobs that are to be replicated.

	<div><div>JSON</div><div><div>Copy</div><pre>{  "properties": {    "policyId": "default",    "sourceAccount": "/subscriptions/&lt;subscriptionId&gt;/resourceGroups/&lt;resource-group&gt;/providers/Microsoft.Storage",    "destinationAccount": "/subscriptions/&lt;subscriptionId&gt;/resourceGroups/&lt;resource-group&gt;/providers/Microsoft.Storage",    "rules": [      {        "ruleId": "",        "sourceContainer": "&lt;source-container&gt;",        "destinationContainer": "&lt;destination-container&gt;",        "filters": {          "prefixMatch": [            "b"          ],          "minCreationTime": "2021-08-02T00:00:00Z"        }      }    ]  }}</pre></div></div>
<div><div>Azure Storage blob inventory</div><div>(create blob inventory)</div></div>	<div><p>The Azure Storage blob inventory feature provides an overview of your containers, blobs, snapshots, and blob versions within a storage account.</p><h3>Inventory features</h3><ul style="list-style-type: none"><li><b>Inventory reports for blobs and containers</b> You can generate inventory reports for blobs and containers. A report for blobs can contain base blobs, snapshots, content length, blob versions and their associated properties such as creation time, last modified time. A report for containers describes containers and their associated properties such as immutability policy status, legal hold status.</li><li><b>Custom Schema</b> You can choose which fields appear in reports. Choose from a list of supported fields. That list appears later in this article.</li><li><b>CSV and Apache Parquet output format</b> You can generate an inventory report in either CSV or Apache Parquet output format.</li></ul></div>

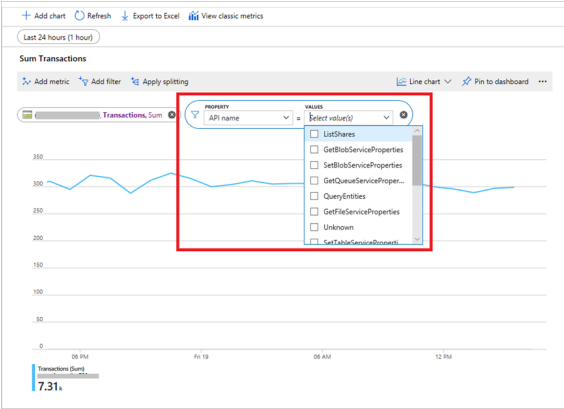
- **Manifest file and Azure Event Grid event per inventory report**  
A manifest file and an Azure Event Grid event are generated per inventory report. These are described later in this article.

Monitoring Azure Blob Storage  
(checkout different blob storage metrics)

Storage Account → Metrics  
This example shows how to view Transactions at the account level.



For metrics that support dimensions, you can filter the metric with the desired dimension value. This example shows how to view Transactions at the account level on a specific operation by selecting values for the API Name dimension.

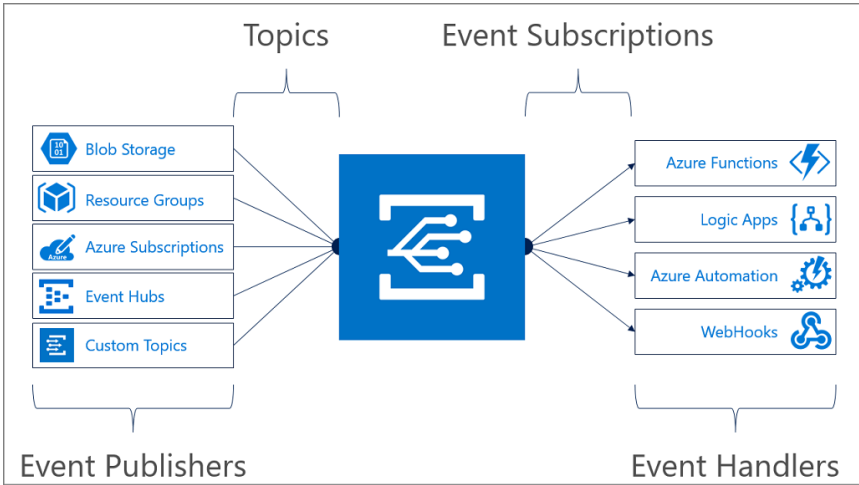
	
<p>Network File System (NFS) 3.0 protocol support for Azure Blob Storage <b>(mount NFS drive and copy few files)</b></p>	<p>Blob storage now supports the Network File System (NFS) 3.0 protocol. This support provides Linux file system compatibility at object storage scale and prices and enables Linux clients to mount a container in Blob storage from an Azure Virtual Machine (VM) or a computer on-premises.</p> <p><b>NFS 3.0 and the hierarchical namespace</b> NFS 3.0 protocol support requires blobs to be organized into a hierarchical namespace. You can enable a hierarchical namespace when you create a storage account. The ability to use a hierarchical namespace was introduced by Azure Data Lake Storage Gen2. It organizes objects (files) into a hierarchy of directories and subdirectories in the same way that the file system on your computer is organized.</p>
<p>SSH File Transfer Protocol (SFTP) support for Azure Blob Storage <b>(use SFTP to upload a file to container)</b></p>	<p>Blob storage now supports the SSH File Transfer Protocol (SFTP). This support lets you securely connect to Blob Storage via an SFTP endpoint, allowing you to use SFTP for file access, file transfer, and file management.</p> <p>SFTP support requires hierarchical namespace to be enabled. Hierarchical namespace organizes objects (files) into a hierarchy of directories and subdirectories in the same way that the file system on your computer is organized.</p> <p><b>SFTP permission model</b> Azure Blob Storage doesn't support Azure Active Directory (Azure AD) authentication or authorization via SFTP. Instead, SFTP utilizes a new form of identity management called local users.</p>

Reacting to Blob storage events  
(subscribe to an event)

Azure Storage events allow applications to react to events, such as the creation and deletion of blobs. Blob storage events are pushed using [Azure Event Grid](#) to subscribers such as Azure Functions, Azure Logic Apps, or even to your own http listener. Event Grid provides reliable event delivery to your applications through rich retry policies and dead-lettering.

The event model

Event Grid uses [event subscriptions](#) to route event messages to subscribers. This image illustrates the relationship between event publishers, event subscriptions, and event handlers.



First, subscribe an endpoint to an event. Then, when an event is triggered, the Event Grid service will send data about that event to the endpoint.

Azure page blobs  
(create page blob and try out few operations on it)

Azure Storage offers three types of blob storage: **Block Blobs**, **Append Blobs** and **page blobs**. **Block blobs** are composed of blocks and are ideal for storing text or binary files, and for uploading large files efficiently.

**Append blobs** are also made up of blocks, but they are optimized for append operations, making them ideal for logging scenarios.

**Page blobs** are made up of **512-byte pages up to 8 TB in total size** and are designed for frequent random read/write operations. Page blobs are the foundation of Azure IaaS Disks. This article focuses on explaining the features and benefits of page blobs.

Page blobs are a collection of 512-byte pages, which provide the ability to read/write arbitrary ranges of bytes. Hence, page blobs are ideal for storing index-based and sparse data structures like OS and data disks for Virtual Machines and Databases. For example, Azure SQL DB uses page blobs as the underlying persistent storage for its databases. Moreover, page blobs are also often used for files with Range-Based updates.

Static website hosting in Azure Storage

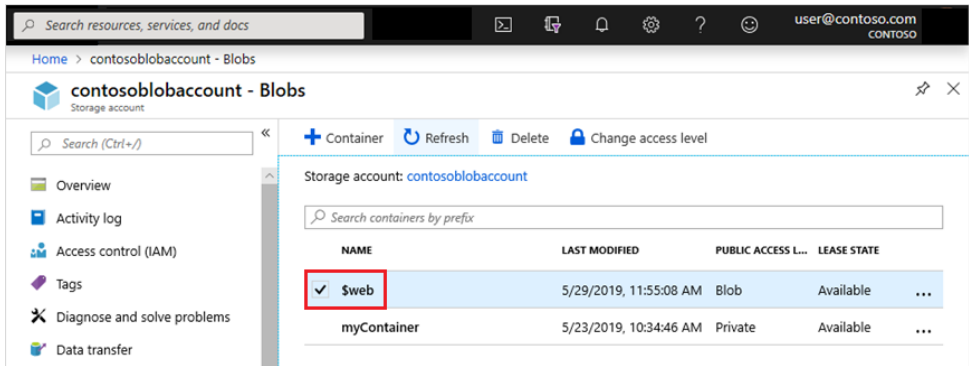
(create a static website, check how to map a domain to it)

You can serve static content (HTML, CSS, JavaScript, and image files) directly from a storage container named \$web.

**Setting up a static website**

Static website hosting is a feature that you have to enable on the storage account.

To enable static website hosting, select the name of your default file, and then optionally provide a path to a custom 404 page. If a blob storage container named \$web doesn't already exist in the account, one is created for you. Add the files of your site to this container.



	<p><b>Viewing content</b></p> <p>The index document that you specify when you enable static website hosting appears when users open the site and don't specify a specific file (For example: <a href="https://contosoblobaccount.z22.web.core.windows.net">https://contosoblobaccount.z22.web.core.windows.net</a>).</p> <p><b>Mapping a custom domain to a static website URL</b></p> <p>You can make your static website available via a custom domain.</p> <p>It's easier to enable HTTP access for your custom domain, because Azure Storage natively supports it. To enable HTTPS, you'll have to use Azure CDN because Azure Storage doesn't yet natively support HTTPS with custom domains. see <a href="#">Map a custom domain to an Azure Blob Storage endpoint</a> for step-by-step guidance.</p> <p>If the storage account is configured to <a href="#">require secure transfer</a> over HTTPS, then users must use the HTTPS endpoint.</p>