Blob snapshots - Azure Storage

By normesta

Archived: 2026-04-05 17:32:02 UTC

A snapshot is a read-only version of a blob that's taken at a point in time.

Note

Blob versioning offers a superior way to maintain previous versions of a blob. For more information, see Blob versioning.

About blob snapshots

A snapshot of a blob is identical to its base blob, except that the blob URI has a **DateTime** value appended to the blob URI to indicate the time at which the snapshot was taken. For example, if a page blob URI is `http://storagesample.core.blob.windows.net/mydrives/myvhd`, the snapshot URI is similar to `http://storagesample.core.blob.windows.net/mydrives/myvhd?snapshot=2011-03-09T01:42:34.9360000Z`.

Note

All snapshots share the base blob's URI. The only distinction between the base blob and the snapshot is the appended **DateTime** value.

A blob can have any number of snapshots. Snapshots persist until they're explicitly deleted, either independently or as part of a Delete Blob operation for the base blob. You can enumerate the snapshots associated with the base blob to track your current snapshots.

When you create a snapshot of a blob, the blob's system properties are copied to the snapshot with the same values. The base blob's metadata is also copied to the snapshot, unless you specify separate metadata for the snapshot when you create it. After you create a snapshot, you can read, copy, or delete it, but you can't modify it.

Any leases associated with the base blob don't affect the snapshot. You can't acquire a lease on a snapshot.

You can create a snapshot of a blob in the hot or cool tier. Snapshots on blobs in the archive tier aren't supported.

A VHD file is used to store the current information and status for a VM disk. You can detach a disk from within the VM or shut down the VM, and then take a snapshot of its VHD file. You can use that snapshot file later to retrieve the VHD file at that point in time and recreate the VM.

Pricing and billing

Creating a snapshot, which is a read-only copy of a blob, can result in extra data storage charges to your account. When designing your application, it's important to be aware of how these charges might accrue so that you can

minimize costs.

Blob snapshots, like blob versions, are billed at the same rate as active data. How snapshots are billed depends on whether you have explicitly set the tier for the base blob or for any of its snapshots (or versions). For more information about blob tiers, see Access tiers for blob data.

If you haven't changed a blob or snapshot's tier, then you're billed for unique blocks of data across that blob, its snapshots, and any versions it may have. For more information, see Billing when the blob tier hasn't been explicitly set.

If you have changed a blob or snapshot's tier, then you're billed for the entire object, regardless of whether the blob and snapshot are eventually in the same tier again. For more information, see Billing when the blob tier has been explicitly set.

For storage accounts that leverage the smart tier public preview, versions and snapshots are billed at full content length. For more information, see Optimize costs with smart tier.

For more information about billing details for blob versions, see Blob versioning.

Minimize costs with snapshot management

Microsoft recommends managing your snapshots carefully to avoid extra charges. You can follow these best practices to help minimize the costs incurred by the storage of your snapshots:

- Delete and re-create snapshots associated with a blob whenever you update the blob, even if you're updating with identical data, unless your application design requires that you maintain snapshots. By deleting and re-creating the blob's snapshots, you can ensure that the blob and snapshots don't diverge.
- If you're maintaining snapshots for a blob, avoid calling methods that overwrite the entire blob when you update the blob. Instead, update the fewest possible number of blocks in order to keep costs low.

Billing when the blob tier hasn't been explicitly set

If you have not explicitly set the blob tier for a base blob or any of its snapshots, then you're charged for unique blocks or pages across the blob, its snapshots, and any versions it may have. Data that is shared across a blob and its snapshots is charged only once. When a blob is updated, then data in a base blob diverges from the data stored in its snapshots, and the unique data is charged per block or page.

When you replace a block within a block blob, that block is later charged as a unique block. This is true even if the block has the same block ID and the same data as it has in the snapshot. After the block is committed again, it diverges from its counterpart in the snapshot, and you'll be charged for its data. The same holds true for a page in a page blob that's updated with identical data.

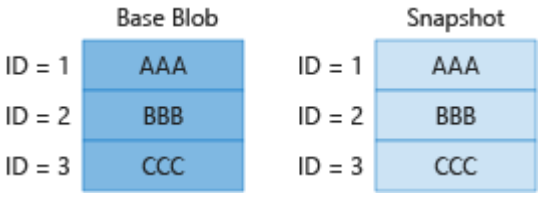
Blob storage doesn't have a means to determine whether two blocks contain identical data. Each block that is uploaded and committed is treated as unique, even if it has the same data and the same block ID. Because charges accrue for unique blocks, it's important to keep in mind that updating a blob when that blob has snapshots or versions results in extra unique blocks and extra charges.

When a blob has snapshots, call update operations on block blobs so that they update the least possible number of blocks. The write operations that permit fine-grained control over blocks are Put Block and Put Block List. The Put Blob operation, on the other hand, replaces the entire contents of a blob and so may lead to extra charges.

The following scenarios demonstrate how charges accrue for a block blob and its snapshots when the blob tier has not been explicitly set.

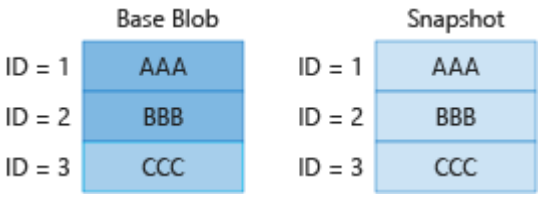
Scenario 1

In scenario 1, the base blob hasn't been updated after the snapshot was taken, so charges are incurred only for unique blocks 1, 2, and 3.



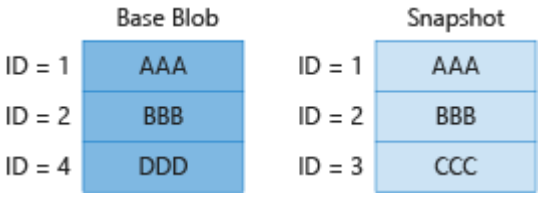
Scenario 2

In scenario 2, the base blob has been updated, but the snapshot hasn't. Block 3 was updated, and even though it contains the same data and the same ID, it isn't the same as block 3 in the snapshot. As a result, the account is charged for four blocks.



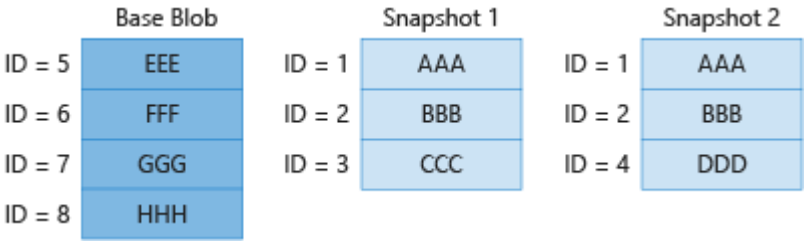
Scenario 3

In scenario 3, the base blob has been updated, but the snapshot hasn't. Block 3 was replaced with block 4 in the base blob, but the snapshot still reflects block 3. As a result, the account is charged for four blocks.



Scenario 4

In scenario 4, the base blob has been completely updated and contains none of its original blocks. As a result, the account is charged for all eight unique blocks.



Tip

Avoid calling methods that overwrite the entire blob, and instead update individual blocks to keep costs low.

Billing when the blob tier has been explicitly set

If you have explicitly set the blob tier for a blob or snapshot (or version), then you're charged for the full content length of the object in the new tier, regardless of whether it shares blocks with an object in the original tier. You're also charged for the full content length of the oldest version in the original tier. Any versions or snapshots that remain in the original tier are charged for unique blocks that they may share, as described in Billing when the blob tier hasn't been explicitly set.

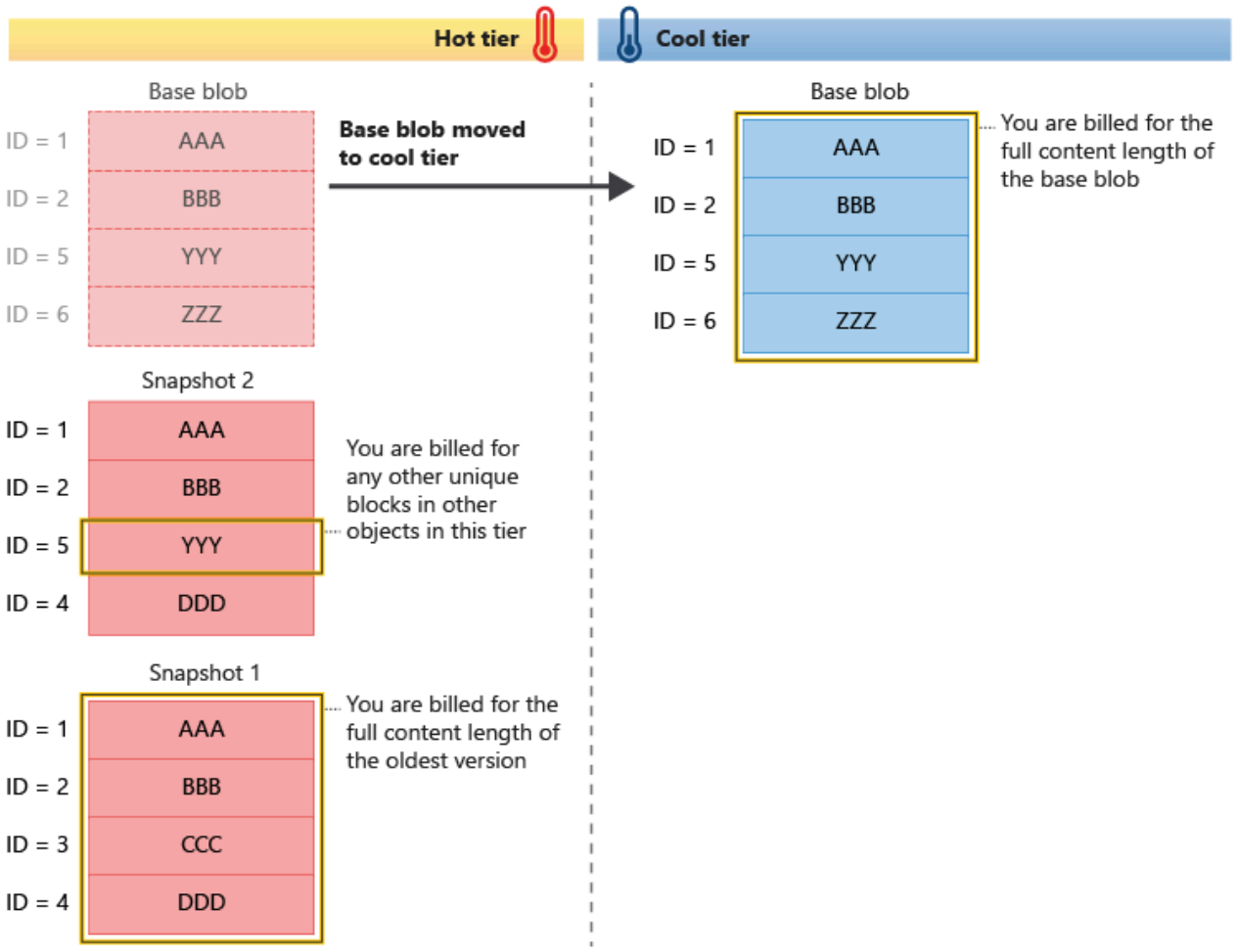
Moving a blob to a new tier

The following table describes the billing behavior for a blob or snapshot when it's moved to a new tier.

When blob tier is set explicitly on...	Then you're billed for...
A base blob with a snapshot	The base blob in the new tier and the oldest snapshot in the original tier, plus any unique blocks in other snapshots.[1]
A base blob with a previous version and a snapshot	The base blob in the new tier, the oldest version in the original tier, and the oldest snapshot in the original tier, plus any unique blocks in other versions or snapshots[1].
A snapshot	The snapshot in the new tier and the base blob in the original tier, plus any unique blocks in other snapshots.[1]

[1]If there are other previous versions or snapshots that haven't been moved from their original tier, those versions or snapshots are charged based on the number of unique blocks they contain, as described in Billing when the blob tier hasn't been explicitly set.

The following diagram illustrates how objects are billed when a blob with snapshots is moved to a different tier.



Explicitly setting the tier for a blob, version, or snapshot can't be undone. If you move a blob to a new tier and then move it back to its original tier, you're charged for the full content length of the object even if it shares blocks with other objects in the original tier.

Operations that explicitly set the tier of a blob, version, or snapshot include:

- Set Blob Tier
- Put Blob with tier specified
- Put Block List with tier specified
- Copy Blob with tier specified

Deleting a blob when soft delete is enabled

When blob soft delete is enabled, if you delete or overwrite a base blob that has had its tier explicitly set, then any previous versions or snapshots of the soft-deleted blob are billed at full content length. For more information about how blob versioning and soft delete work together, see Blob versioning and soft delete.

The following table describes the billing behavior for a blob that is soft-deleted, depending on whether versioning is enabled or disabled. When versioning is enabled, a new version is created when a blob is soft-deleted. When versioning is disabled, soft-deleting a blob creates a soft-delete snapshot.

When you overwrite a base blob with its tier explicitly set...	Then you're billed for...
If blob soft delete and versioning are both enabled	All existing versions at full content length regardless of tier.
If blob soft delete is enabled but versioning is disabled	All existing soft-delete snapshots at full content length regardless of tier.

Feature support

Support for this feature might be impacted by enabling Data Lake Storage Gen2, Network File System (NFS) 3.0 protocol, or the SSH File Transfer Protocol (SFTP). If you've enabled any of these capabilities, see Blob Storage feature support in Azure Storage accounts to assess support for this feature.

Important

The Snapshots Preview for accounts that have the hierarchical namespace feature enabled is no longer accepting new customers. We encourage you to consider alternative mechanisms. (Examples: Soft delete for blobs, AzCopy, Vaulted Backup [Preview])

Next steps

- Blob versioning
- Create and manage a blob snapshot in .NET
- Back up Azure unmanaged VM disks with incremental snapshots

---