

Azure Storage Account

Storage Account

Blob, file, table, queue

Data Lake

Snapshot backup to Azure blob storage

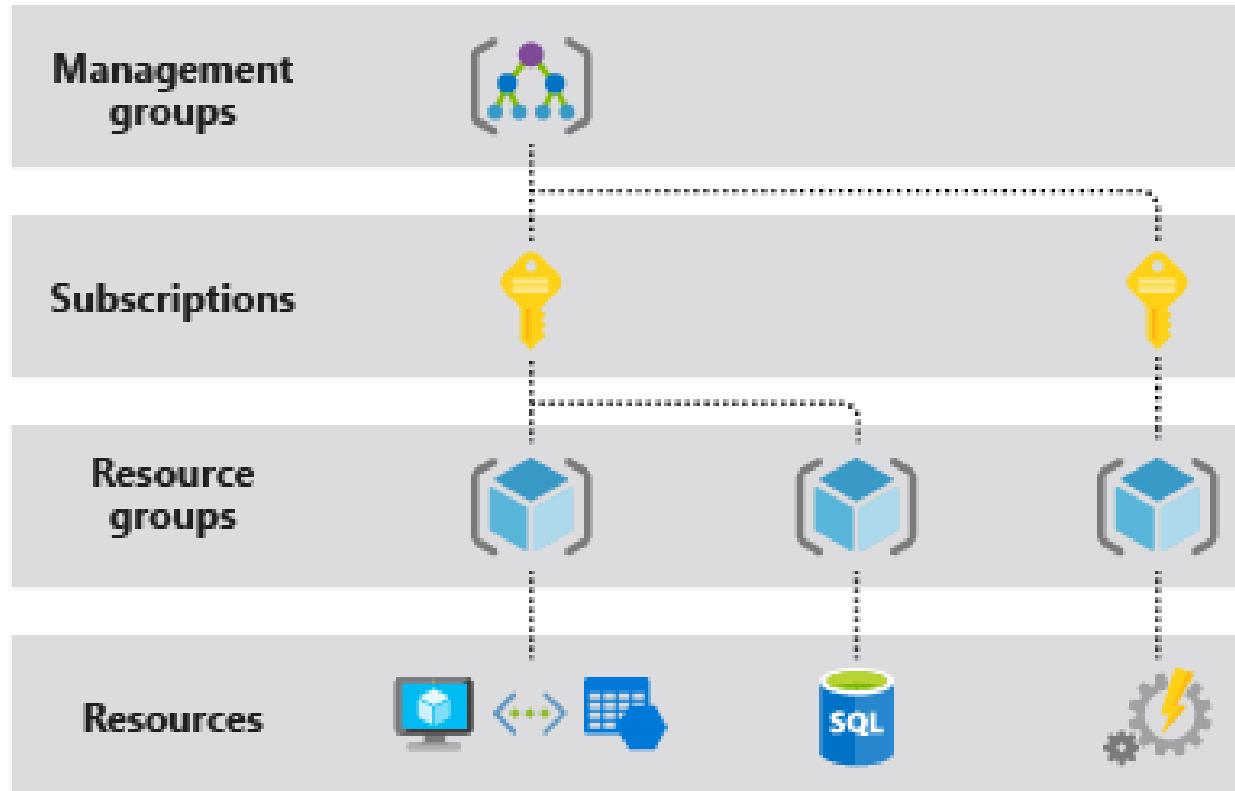
May 2022 Camilla Gaardsted

SuperUsers A/S

Data centers geography



Azure Resources



Resources are arranged in resource groups

Resources in this course:

- **Azure Storage Account**
- Azure Databricks
- Azure Synapse Analytics
- Azure Data Factory
- Azure EventHub
- Azure Stream Analytics job
- Azure CosmosDB

Azure common tools

Online portal portal.azure.com (nice GUI)

Online **shell** shell.azure.com (bash/powershell)

- Azure CLI
- PowerShell az module

PowerShell locally/online with az module

- Direct cmdlets
- Via templates (from portal/github)

Azure CLI locally/online (cross platform cmd line interface)

Azure Storage Explorer (cross platform GUI file/storage explorer tool)

Visual Studio Code (free cross platform text editor)

Azure Storage account

Azure storage via HTTP/HTTPS

A storage account must have a **global unique name** e.g.
"superusersstorage2022"


- suffix is .core.windows.net dvs: superusersstorage2022.core.windows.net
- Name is 3-24 characters. Only lower case letters and numbers are allowed.

All objects in a storage account are billed together as a group. Objects are related to the 4 **services**:

Data storage

 Containers

 File shares

 Queues

 Tables

Azure storage **services** – we have 4 types

- Blob
- File share
- Queue
- Table

No OS to worry about – it's just services!

Storage **account kind** determines which of the **4** we have access to

- General purpose v2 gives access to all 4 but...

Storage Account – Account Types

Account type	Service(s)	Redundancy Options	In portal	Data Lake option
Standard General purpose v2	All 4	LRS, ZRS,[RA-]GRS,[RA-]GZRS	Yes	Yes
Premium General purpose v2	Blob only	LRS, ZRS	No	No
Premium block blobs	Blob	LRS, ZRS	Yes	Yes
Premium page blobs	Blob	LRS	Yes	No
Premium file shares	File share	LRS, ZRS	Yes	No

All 4: Blob, File, Queue, Table service

All other standard account types are considered legacy.

RA means Read Access to secondaries

In portal: Available in portal.azure.com under storage account creation

Storage Account – Pricing for a Data Lake

Pay-as-you-go or get discount for a 1 or 3 year reserved capacity

	Premium	Hot	Cool	Archive
First 50 terabyte (TB) / month	\$0.195 per GB	\$0.02 per GB	\$0.01 per GB	\$0.0018 per GB
Next 450 TB / month	\$0.195 per GB	\$0.0188 per GB	\$0.01 per GB	\$0.0018 per GB
Over 500 TB / month	\$0.195 per GB	\$0.018 per GB	\$0.01 per GB	\$0.0018 per GB

Operations and data transfer

	Premium	Hot	Cool	Archive
Write operations (per 10,000) ¹	\$0.0296	\$0.07	\$0.13	\$0.156
Read operations (per 10,000) ²	\$0.0024	\$0.006	\$0.013	\$7.80
Iterative Read Operations (per 10,000) ³	N/A	\$0.006	\$0.013	\$7.80
Iterative Write Operations (100's) ⁴	N/A	\$0.07	\$0.13	\$0.156
Data Retrieval (per GB)	N/A	N/A	\$0.01	\$0.024
Data Write (per GB)	Free	Free	Free	Free
Index (GB/month)	N/A	\$0.028	N/A	N/A
All other Operations (per 10,000), except Delete, which is free	\$0.0024	\$0.006	\$0.013	\$7.80

¹ The following API calls are considered write operations: AppendFile, CreateFilesystem, CreatePath, CreatePathFile, FlushFile, SetFileProperties, SetFilesystemProperties, RenameFile, RenamePathFile, CopyFile

² The following API calls are considered read operations: ReadFile, ListFilesystemFile

³ The following API calls are considered iterative read operations: List Filesystem & List Path

⁴ The following API calls are considered iterative write operations: RenameDirectory, RenamePath, RenamePathDir

Create Storage Account via Azure Portal

Storage account name ⓘ *

Region ⓘ *

Performance ⓘ *
☐ Standard: Recommended for most scenarios (general-purpose v2 account)
☒ Premium: Recommended for scenarios that require low latency.

Premium account type ⓘ *

Redundancy ⓘ *

NB Standard/Premium performance choice determines possibilities for these dependant options:

- Account kind
- Redundancy (Standard has most options)
- Default Access tier (standard blob only)
- Hierarchical namespace

Storage account - Performance

Two possibilities at creation (**cannot be changed later**):

- Standard (HDD –big data/bulk storage/infrequently used data)
- Premium (SSD – low latency, I/O intensive e.g databases)

Storage account - Redundancy

99,9% of a
year is 8.77
hours

99,99% of
a year is 53
minutes

Redundancy setting at account level for all services

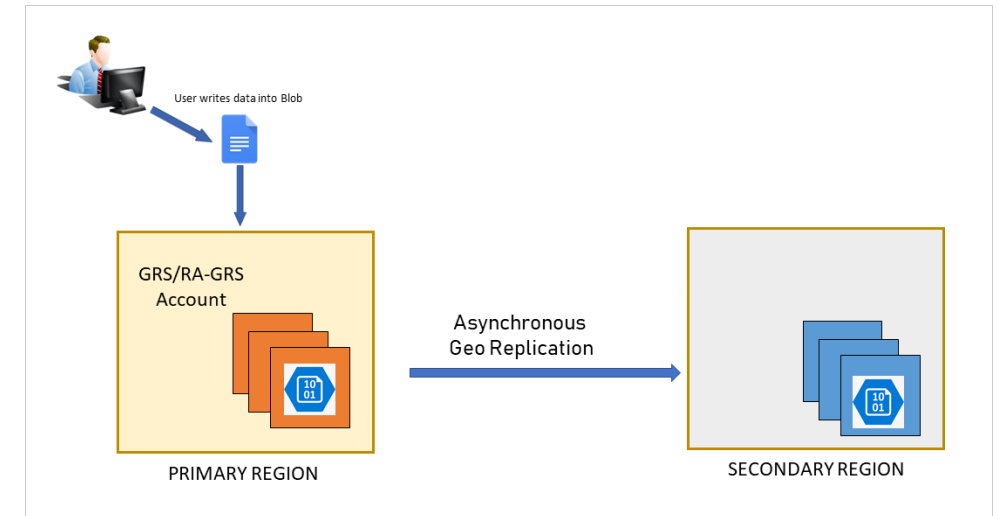
Type	% Durability over a year	#Copies	Read Availability	Write Availability
LRS	99.999999999% (11 9's)	3 in a single region	99,9%	99,9%
ZRS	99.9999999999% (12 9's)	3 across zones in region	99,9%	99,9%
GRS	99.99999999999999% (16 9's)	3 in primary region + 3 in secondary region	99,9%*	99,9%*
RA-GRS	99.99999999999999% (16 9's)	3 in primary region + 3 in secondary region	99,99%*	99,9%*
GZRS	99.99999999999999% (16 9's)	3 across zones in region + 3 in secondary region	99,9%*	99,9%*
RA-GZRS	99.99999999999999% (16 9's)	3 across zones in region + 3 in secondary region	99,99%*	99,9%*

**Cool and archive access tiers have one less 9 cipher here*

Replication – ønske om Redundancy

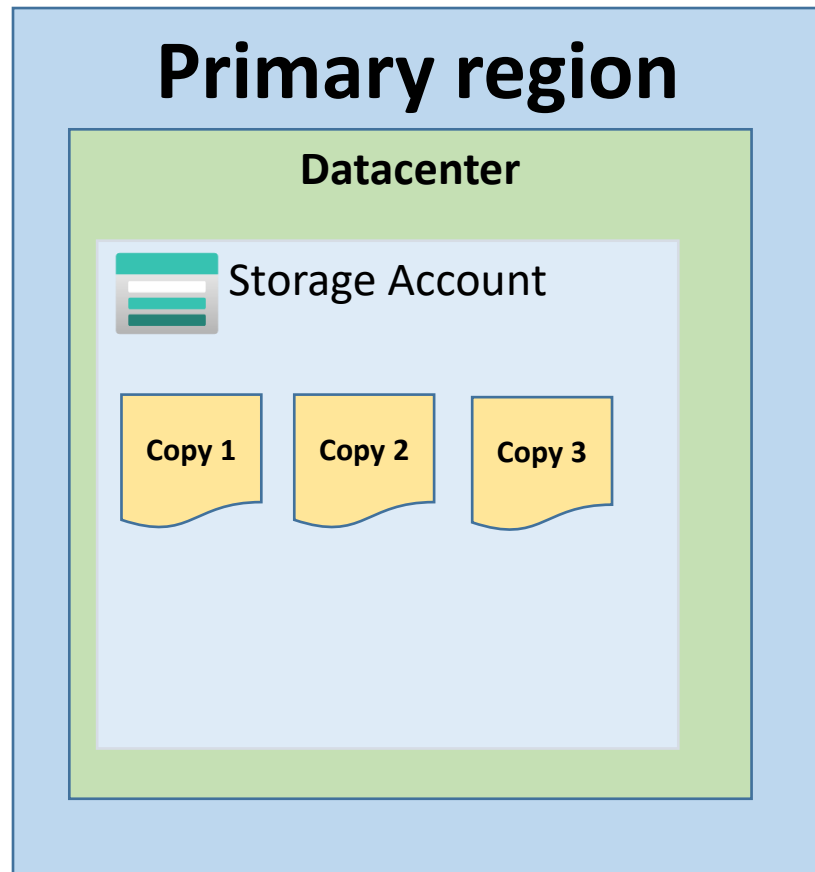
Primary and eventually a secondary region

- Locally-redundant storage (LRS)
- Zone-redundant storage (ZRS)
- Geo-redundant storage (GRS)
- Read-access-geo-redundant storage (RA-GRS)
- Geo-redundant (GZRS)
- Read-access-geo-zone-redundant (RA-GZRS)



- <https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy>
- You can choose to replicate your data within the same data center, across zonal data centers within the same region, and even across regions.

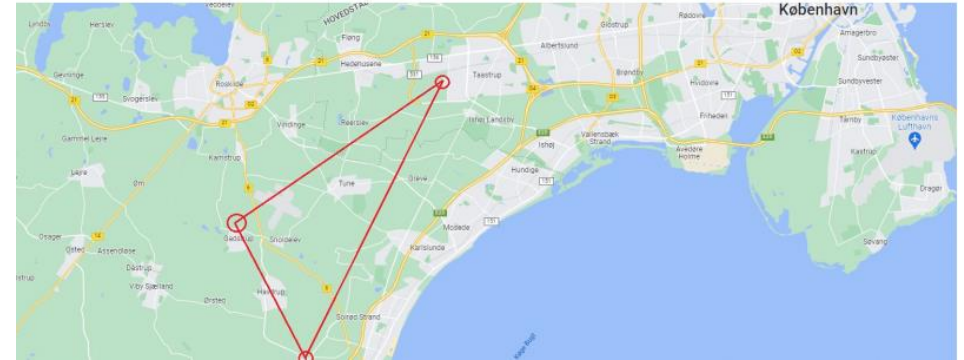
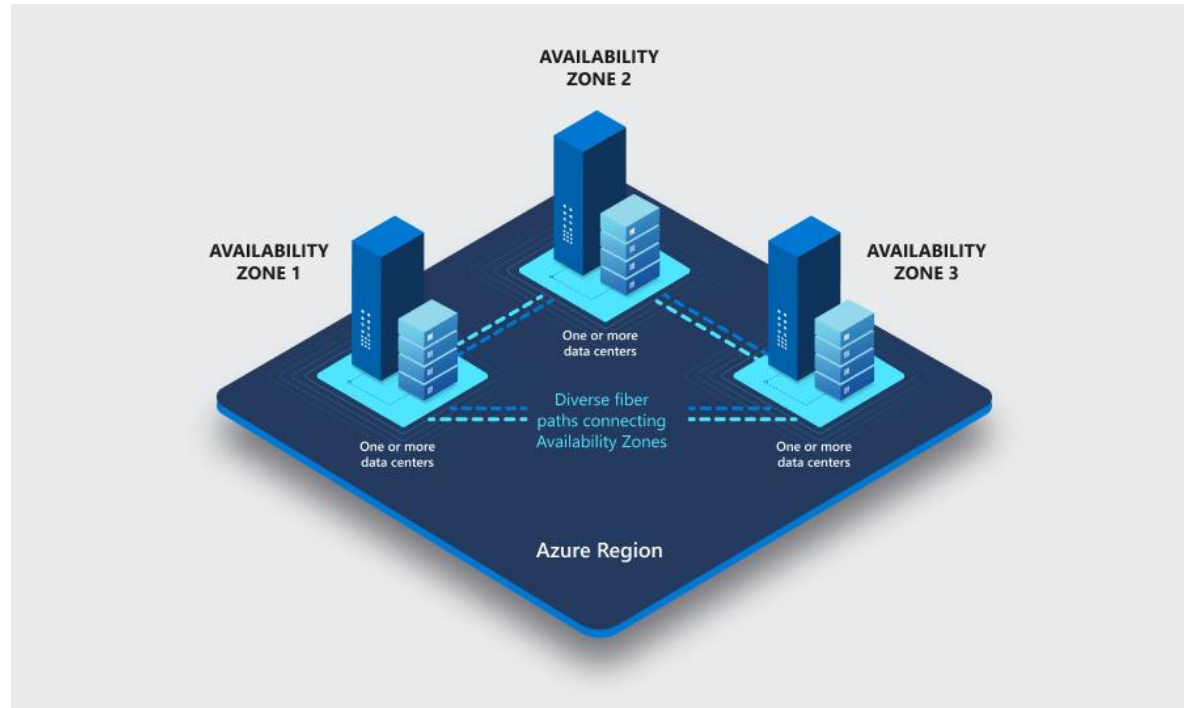
Redundancy – replication – LRS - synkron



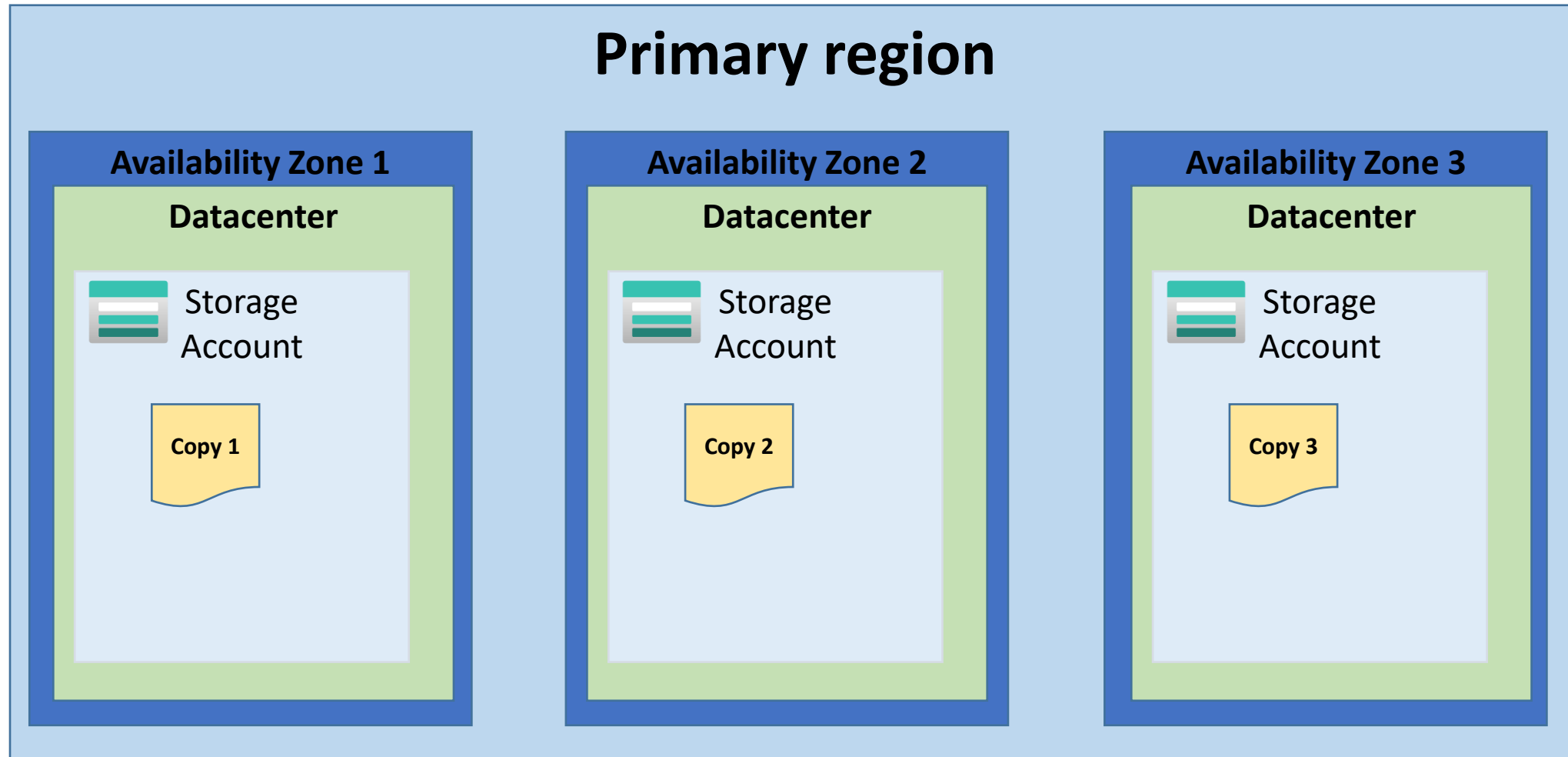
Replication – fault and update domain

- A write request to an LRS storage account returns successfully only after the data has been written to all 3 replicas. These replicas each reside in **separate fault domains and update domains** within one storage scale unit.
- A storage scale unit is a collection of racks of storage nodes.
 - A fault domain (FD) is a group of nodes that represent a **physical unit** of failure and can be considered as nodes belonging to the same physical rack.
 - An **upgrade domain** (UD) is a group of nodes that are upgraded together during the process of a service upgrade (rollout). The replicas are spread across UD's and FD's within one storage scale unit. This architecture ensures that your data is available if a hardware failure impacts a single rack or when nodes are upgraded during a rollout.

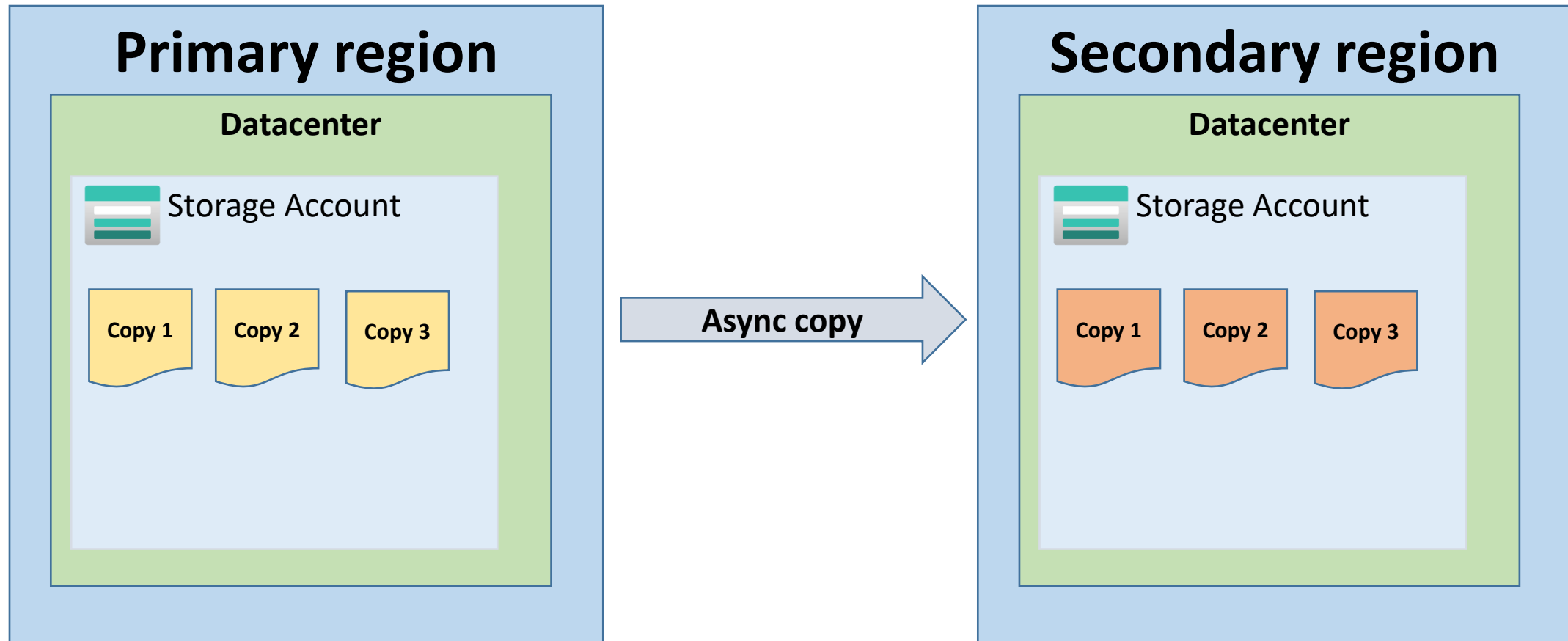
Azure Availability Zones



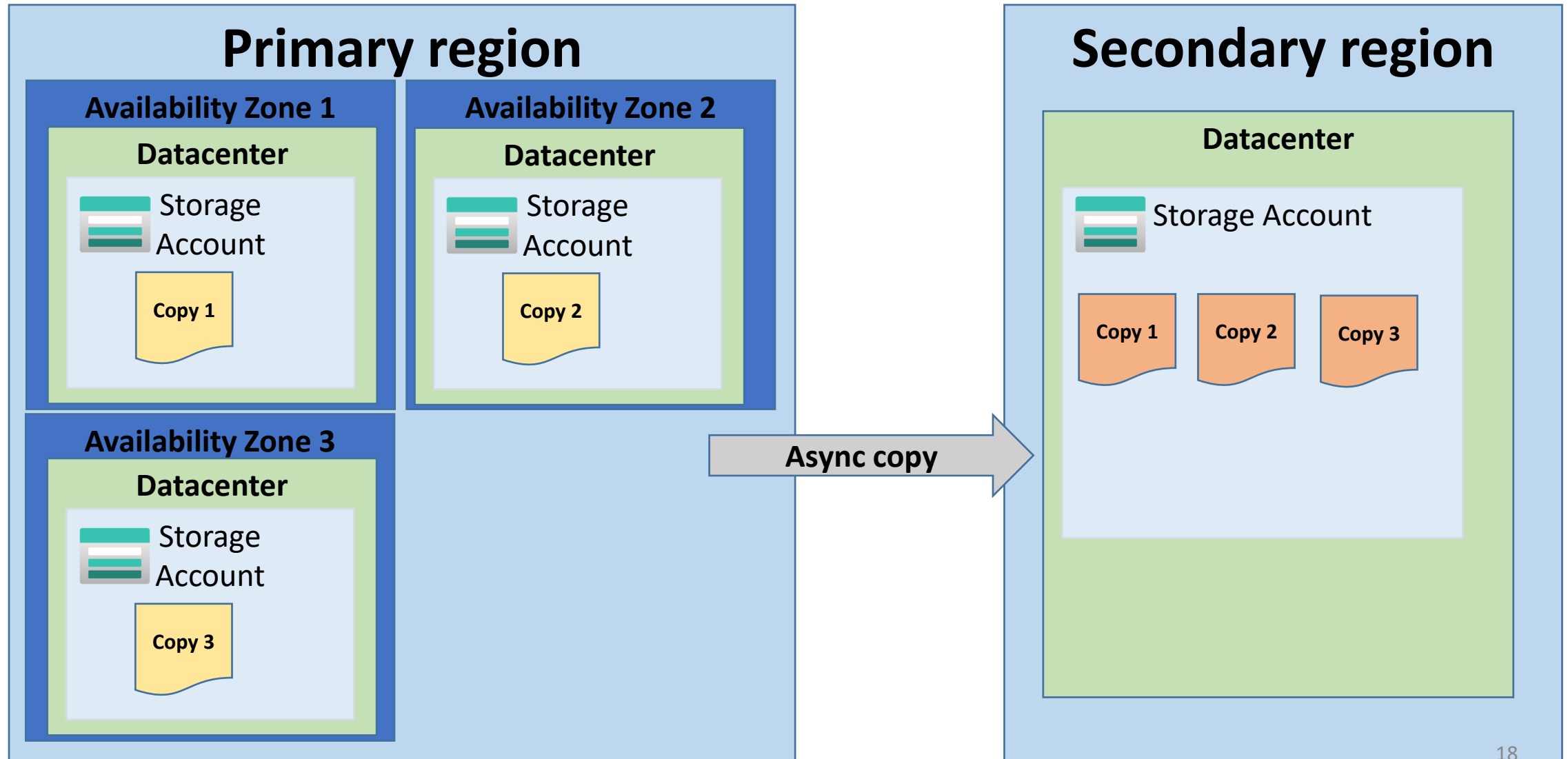
Redundancy – replication – ZRS - synkron



Redundancy – replication – GRS



Redundancy – replication – GZRS



Azure Geo replication – paired regions



Geography	Regional pair A	Regional pair B
Europe	North Europe (Ireland)	West Europe (Netherlands)
Germany	Germany West Central	Germany North*
Norway	Norway East	Norway West*
Sweden	Sweden Central	Sweden South*

Redundancy –RA-GRS

Secondary is available for read

Endpoints

Primary endpoint

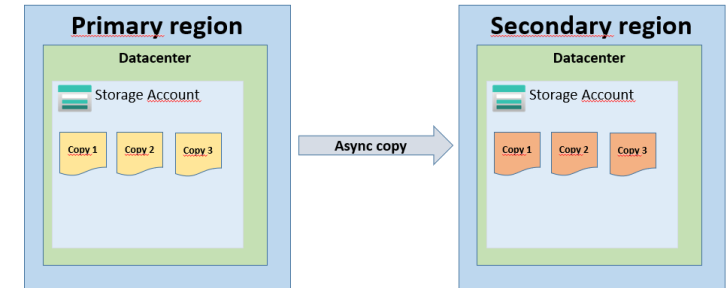
Blob service

<https://premstorage20220511.blob.core.windows.net/>

Secondary endpoint

Blob service

<https://premstorage20220511-secondary.blob.core.windows.net/>



Microsoft-managed failover

In extreme circumstances where a region is lost due to a significant disaster, Microsoft may initiate a regional failover. In this case, no action on your part is required. Until the Microsoft-managed failover has completed, you won't have write access to your storage account. Your applications can read from the secondary region if your storage account is configured for RA-GRS or RA-GZRS.

Manual failover (not for Data Lake) gives LRS

- Your last sync time is **5/11/2022, 9:56:17 AM**. You may lose any data after this time if you initiate the failover.
- After the failover of a GRS or RA-GRS account, your storage account replication will be converted to a locally-redundant storage(LRS) account. You can convert your account to use GRS or RA-GRS.

Storage account – storage replication

Outage scenario	LRS	ZRS	GRS/RA-GRS	GZRS/RA-GZRS
A node within a data center becomes unavailable	Yes	Yes	Yes	Yes
An entire data center (zonal or non-zonal) becomes unavailable	No	Yes	Yes ¹	Yes
A region-wide outage occurs in the primary region	No	No	Yes ¹	Yes ¹
Read access to the secondary region is available if the primary region becomes unavailable	No	No	Yes (with RA-GRS)	Yes (with RA-GZRS)

¹ Account failover is required to restore write availability if the primary region becomes unavailable. For more information, see [Disaster recovery and storage account failover](#).

Source: <https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy>

Default Access tier – **Standard** blob only



- Only for **Standard** Blob storage e.g:
 - General-purpose-V2
 - BlobStorage account (legacy)

Access Tier	When	Set as default
Hot	Frequently used data	Yes
Cool	Infrequently used data	Yes
Archive	Rarely accessed	No

Access tier can be **set individually** on a given blob and changed later

Choose default access tier (inferred) on account creation

3rd possibility **Archive** (lowest storage costs, highest access cost and **rehydration takes up to 15 hours!**)

Name	Modified	Access tier
<input type="checkbox"/>  weatherstationdata.csv	5/11/2022, 8:39:54 AM	Hot
<input type="checkbox"/>  weatherstations.csv	5/10/2022, 12:30:14 PM	Cool (Inferred)

Storage Account – Changing redundancy

LRS to GRS and vice versa – no problem just change in GUI

LRS to ZRS – manual or request live migration from ms

Storage Account - Data Lake Gen2 upgrade

A standard Gen2 storage account can be upgraded to a Data Lake Gen2

Settings

- Configuration
- Data Lake Gen2 upgrade**
- Resource sharing (CORS)
- Advisor recommendations
- Endpoints
- Locks

^ ① Step 1: Review account changes before upgrading - Not started

A number of data protection features, along with other features, will need to be disabled due to conflicts with the upgrade.

Review and agree to changes

^ ② Step 2: Validate account before upgrading - Not started

All features that are not supported with Azure Data Lake Storage Gen2 will be checked during validation, and a full list of discrepancies (if any) will be available as a blob in a generated container. This may take multiple attempts.

Start validation

i Can't start validation until step 1 is completed

∨ ③ Step 3: Upgrade account - Not started

Unsupported blob types

- Page blob

Unsupported data protection

- Container snapshot
- Container soft delete
- Point-in-time restore

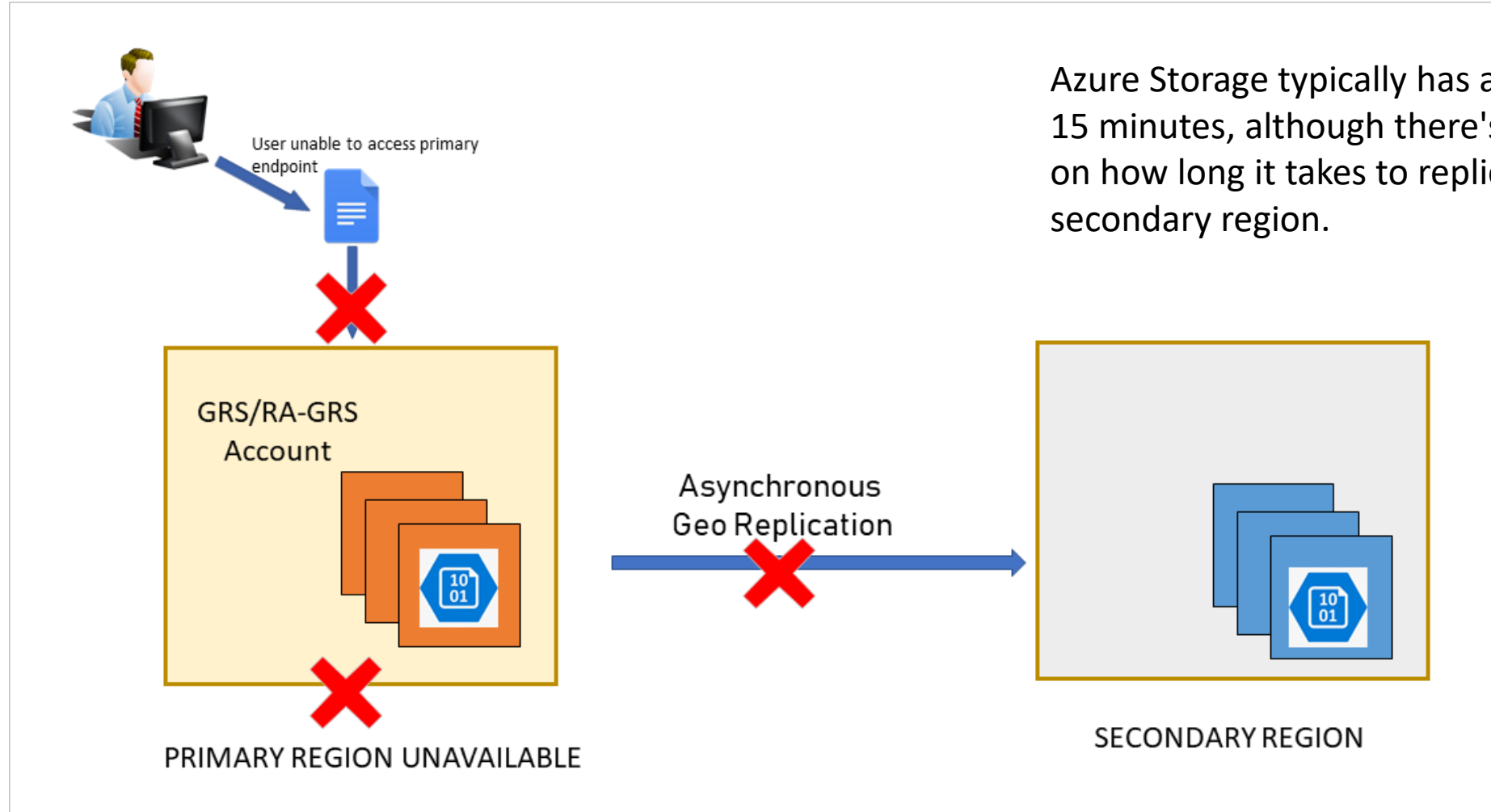
Other unsupported capabilities

- Change feed
- Active leasing
- Blob tagging
- Container rename
- Customer-provided key (CPK)
- Encryption scope

Migrating to another replication type

Switching	...to LRS	...to GRS/RA-GRS	...to ZRS	...to GZRS/RA-GZRS
...from LRS	N/A	Use Azure portal, PowerShell, or CLI to change the replication setting ¹	Perform a manual migration Request a live migration	Perform a manual migration OR Switch to GRS/RA-GRS first and then request a live migration ¹
...from GRS/RA-GRS	Use Azure portal, PowerShell, or CLI to change the replication setting	N/A	Perform a manual migration OR Switch to LRS first and then request a live migration	Perform a manual migration Request a live migration
...from ZRS	Perform a manual migration	Perform a manual migration	N/A	Use Azure portal, PowerShell, or CLI to change the replication setting ¹
...from GZRS/RA-GZRS	Perform a manual migration	Perform a manual migration	Use Azure portal, PowerShell, or CLI to change the replication setting	N/A

Disaster recovery and account failover



Azure Data Lake

A set of capabilities dedicated to **big data analytics**, built on **Azure Blob Storage**

Azure Data Lake - Creation

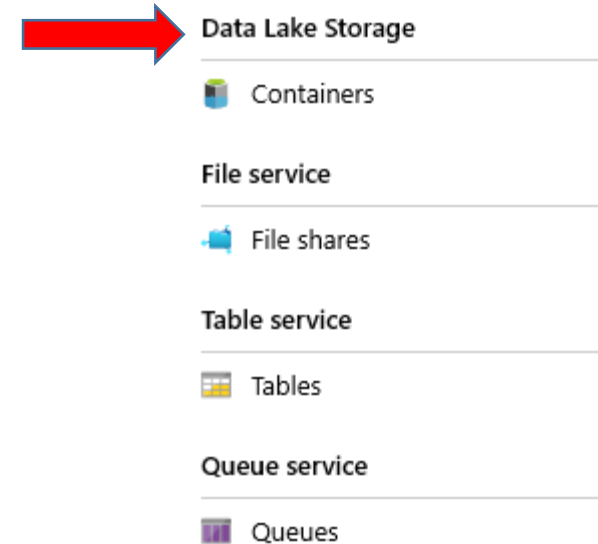


Hierarchical namespace

Enabled

- Data Lake Storage Gen1 (**legacy**) had it's own resource
- **Data Lake Storage Gen2** (recommended)
 - Is just a tiny checkbox on a **Standard v2** storage account under Advanced
 - Blob service section renamed to Data Lake
 - **Almost all** benefits from blob storage

Known as **ADSL Gen2**



Azure Data Lake Gen2

Hierarchical namespace

- The hierarchical namespace organizes objects/files into a hierarchy of directories for efficient data access
- Rename/delete directory is a metadata operation
- Search – can skip irrelevant paths

Hadoop compatible access

- Manage and access data just as you would with a Hadoop Distributed File System (HDFS)

Avoid many small files (100MB to 1GB is optimal)

Partition data in folders used in common filters of the data (year/month/ , country/customer/ etc)

Blob or Azure Data Lake Gen2?

Blobs are stored as objects in one single flat namespace e.g. the slash is just part of the name

- data/2021/januarysale.csv

Filesystem in a data lake is hierarchial:

- Data
 - 2021
 - Januarysale.csv

Storage account – Network

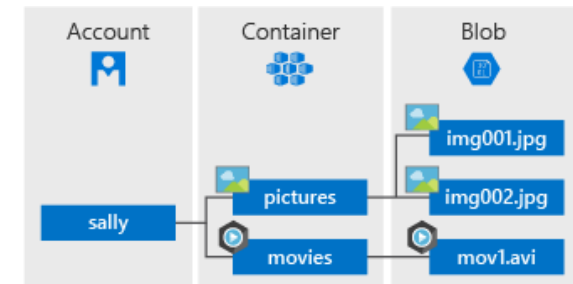
- Public network access
- Selected virtual networks
- Selected IP addresses (firewall rules)

Storage account - backup

- Replication is not enough!
 - Application errors
 - Hacker attack
 - Accidental deletion
- No standard solution yet for a complete storage account
- Azure Backup service can backup Azure File Shares (preview)
- Blogpost about blobstorage backup
- New point in time recovery for container

Blob storage

- Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of **unstructured 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.
- Objects in Blob storage can be accessed from anywhere in the world via HTTP or HTTPS. Users or client applications can access blobs via URLs, the [Azure Storage REST API](#), [Azure PowerShell](#), [Azure CLI](#), or an Azure Storage client library.
- The storage client libraries are available for multiple languages, including [.NET](#), [Java](#), [Node.js](#), [Python](#), [PHP](#), and [Ruby](#).



Azure blob storage

- **Block blobs** store text and binary data, up to about 4.7 TB. Block blobs are made up of blocks of data that can be managed individually.
- **Append blobs** are made up of blocks like block blobs, but are optimized for append operations. Append blobs are ideal for scenarios such as logging data from virtual machines.
- **Page blobs** store random access files up to 8 TB in size. Page blobs store virtual hard drive (VHD) files and serve as disks for Azure virtual machines. For more information about page blobs, see [Overview of Azure page blobs](#)

Azure file share service

Azure File Share service enables you to set up highly available **network file shares** that can be accessed by using Server Message Block (SMB) protocol.

- No OS for you to worry about
- Multiple VMs can share the same files with both read and write access.
- You can also read the files using the REST interface or the storage client libraries.
- Can be mounted on-premise eg. Windows, Linux, Mac

Azure file share

- To mount an Azure file share, you will need the primary (or secondary) **storage account key**. SAS keys are not currently supported for mounting.
- One thing that distinguishes Azure Files from files on a corporate file share is that you can access the files from anywhere in the world using a URL that points to the file and includes a shared access signature (SAS) token. You can generate SAS tokens; they allow specific access to a private asset for a specific amount of time.

Azure file shares

Premium

- Only via FileStorage kind
- Billed for Provisioned size – no quota choice

Standard

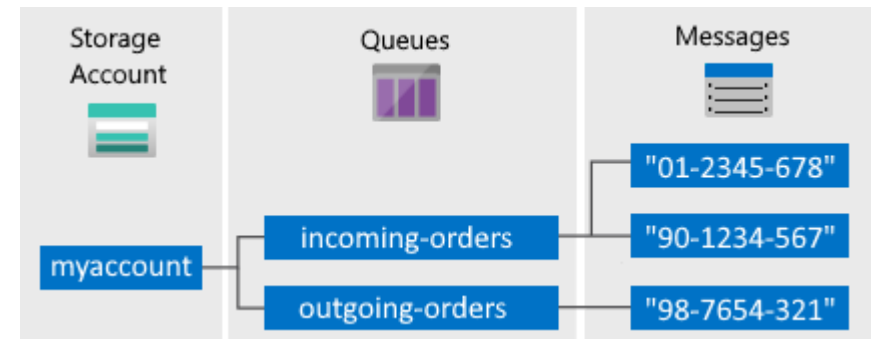
- Only via Gpv2 kind
- Qouta choice is upper boundary of file share

Queue storage

- The Azure Queue service 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 generally used to store lists of messages to be processed asynchronously.
- For example, say you want your customers to be able to upload pictures, and you want to create thumbnails for each picture. You could have your customer wait for you to create the thumbnails while uploading the pictures. An alternative would be to use a queue. When the customer finishes his upload, write a message to the queue. Then have an Azure Function retrieve the message from the queue and create the thumbnails. Each of the parts of this processing can be scaled separately, giving you more control when tuning it for your usage.

Queue Storage

- Queue storage delivers asynchronous messaging between application components, whether they are running in the cloud, on the desktop, on an on-premises server, or on a mobile device
- Storage account means https/http
- Can contain millions of messages
- Queue name is lowercase

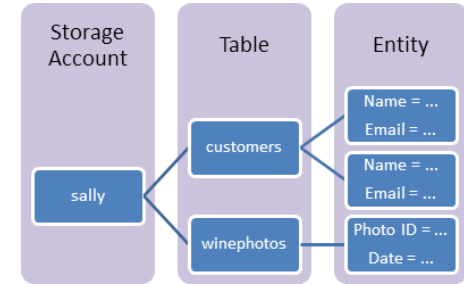


Queue Storage

A message has a time-to-live

- Default is 7 days
- -1 means it never expires

Table storage



- Azure Table storage is now part of Azure Cosmos DB.
- Azure Table storage stores large amounts of structured data. The service is a NoSQL datastore which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of Table storage include:
- Storing TBs of structured data capable of serving web scale applications
- Storing datasets that don't require complex joins, foreign keys, or stored procedures and can be denormalized for fast access
- Quickly querying data using a clustered index
- Accessing data using the OData protocol and LINQ queries with WCF Data Service .NET Libraries
- You can use Table storage to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

Table storage

Table: A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties.

Entity: An entity is a set of properties, similar to a database row. An entity in Azure Storage can be up to 1MB in size. An entity in Azure Cosmos DB can be up to 2MB in size.

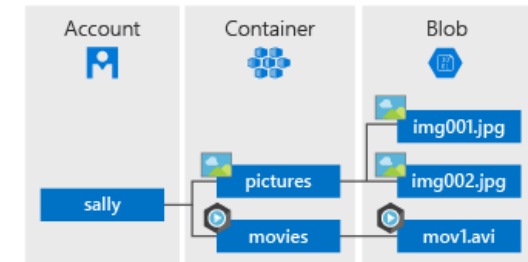
Properties: A property is a name-value pair. Each entity can include up to 252 properties to store data

Each entity also has three system properties that specify a partition key, a row key, and a timestamp

PartitionKey must be string

RowKey together with PartitionKey must be unique

Blob service Containers



- A container organizes a set of blobs, **similar to a folder** in a file system. All blobs reside within a container. A storage account can contain an unlimited number of containers, and a container can store an unlimited number of blobs.
- **Container name must be lowercase.**
- Container names must start with a letter or number, and can contain only letters, numbers, and the dash (-) character.
- Container names must be from 3 through 63 characters long.

Create container

- Provide a lowercase name
- Decide Public access level:
 - Private (no anonymous access)
 - Blob (public access via url to read blobs in the container)
 - Containers (public access via url to enumerate and read blobs in the container)

Enumerate via REST API:

`https://<storageaccount>.blob.core.windows.net/<container>?restype=container&comp=list`

Blob types

- All blobs reside within a container. You can further organize blobs into **virtual directories**, and traverse them as you would a file system.
- 3 blob types:
 - Block blobs store text and binary data, up to about 4.7 TB. Block blobs are made up of blocks of data that can be managed individually.
 - Append blobs are made up of blocks like block blobs, but are optimized for append operations. Append blobs are ideal for scenarios such as logging data from virtual machines.
 - Page blobs store random access files up to 8 TB in size. Page blobs store the VHD files that back VMs.
- For very large datasets where network constraints make uploading or downloading data to Blob storage over the wire unrealistic, you can **ship a set of hard drives to Microsoft** to import or export data directly from the data center.

Storage Account - Tools

Azure Storage Explorer

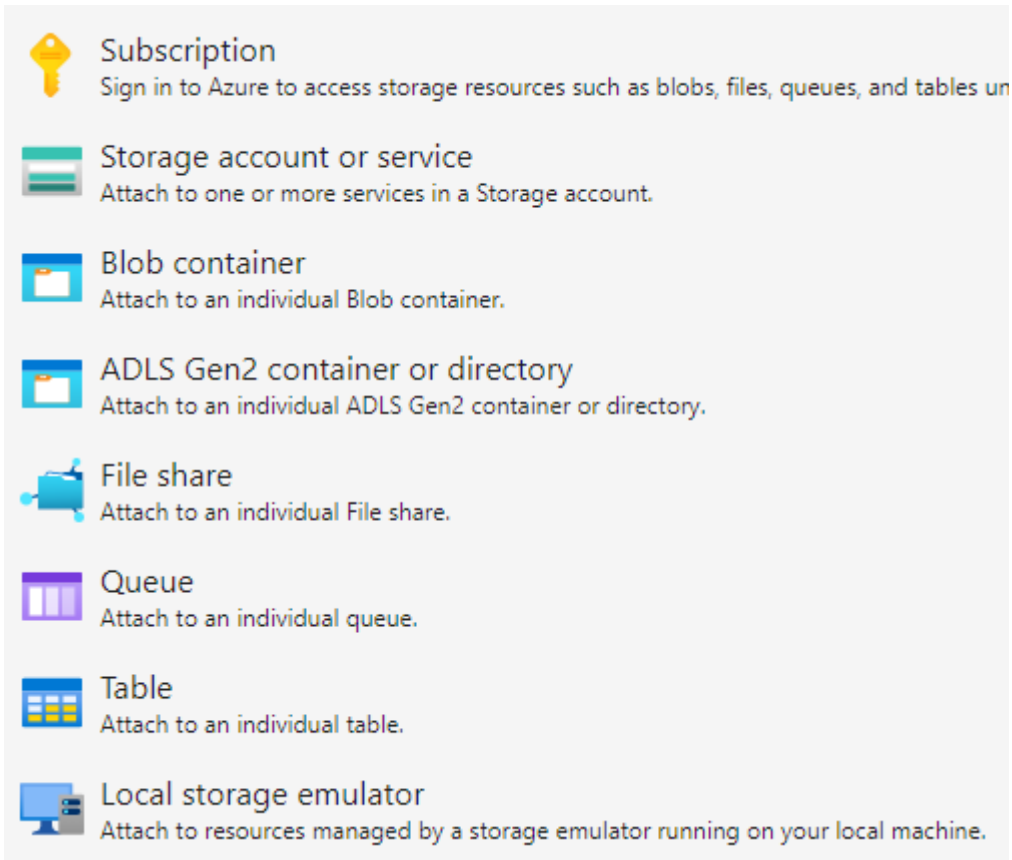
- Free GUI client tool (cross platform) uses AzCopy under the hood
- Download from <https://azure.microsoft.com/en-us/features/storage-explorer/>
- Upload, download, and manage containers, blobs, files, queues, tables, and Cosmos DB entities. Gain easy access to manage your virtual machine disks.

Storage Account – AzCopy

AzCopy is a free command line utility (cross platform) to copy blobs/files to/from a storage account

Storage type	Currently supported method of authorization
Blob storage	Azure AD & SAS
Blob storage (hierarchical namespace)	Azure AD & SAS
File storage	SAS only

Azure Storage Explorer – Connect to



- Connect via
 - Azure account
 - Azure AD
 - Connectionstring
 - SAS -Shared access signature
 - Storage account name + key
 - Local emulator

Storage Account - Security

Storage account keys (access keys)

- storage account key is similar to the root password for your storage account (complete access).
- You are provided **two access keys** so that you can maintain connections using one key while regenerating the other.
- Storage account keys are 512-bit strings created by Azure that, along with the storage account name, can be used to access the data objects stored in the storage account, for example, blobs, entities within a table, queue messages, and files on an Azure file share. Controlling access to the storage account keys controls access to the data plane for that storage account.

Storage account keys – connect

Microsoft Azure Storage Explorer - Connect

×

Attach using Name and Key

Enter information to connect to the Microsoft Azure storage account

Account name:
gaardsted20180613

Account key:
irjYcE4Q2tEjFXbO/LElFDaNUPcbXBjNSN6ftuWac2SISDscC1YeUtbCMHkP3emGksF09HS+7qeMF746MdXGcQ==

Storage endpoints domain:
Azure

☐ Use HTTP (Not recommended)
[Online privacy statement](#)

Storage account name
gaardsted20180613

key1
Key
irjYcE4Q2tEjFXbO/LElFDaNUPcbXBjNSN6ftuWac2SISDscC1YeUtbCMHkP3emGksF09HS+7qeMF746MdXGcQ==

Connection string
DefaultEndpointsProtocol=https;AccountName=gaardsted20180613;AccountKey=irjYcE4Q2tEjFXbO/LElFDaNUPcbXBjNSN6ftuWac2SISDscC1Ye...

key2
Key
cl8EvYzUMyfrMleOCB+VzQ1oxQSGAXW0DoUC/tzklzS4puOz7luPGEzu/xKyjksekEuRycZZqTmUX5FaV85WUA==

Connection string
DefaultEndpointsProtocol=https;AccountName=gaardsted20180613;AccountKey=cl8EvYzUMyfrMleOCB+VzQ1oxQSGAXW0DoUC/tzklzS4puOz7...

Storage permissions - RBAC




Access Control (IAM)

Identity and Access Management

For Azure AD we have Role based access control (RBAC) built in roles in Azure at **scopes** at **container, storage account and resource group level**:

- Storage Blob Data Reader
- **Storage Blob Data Contributor**
- Storage Blob Data Owner
- Storage Blob Delegator

<input type="checkbox"/>	Name	Type	Role	Scope
✓	Storage Blob Data Contributor			
<input type="checkbox"/>	 BI Group	Group	Storage Blob Data Contributor ⓘ	This resource

- Read and list Azure Storage containers and blobs.
- Read, write, and delete Azure Storage containers and blobs.
- Provides full access to Azure Storage blob containers and data, including assigning POSIX access control.
- Get a user delegation key, which can then be used to create a shared access signature for a container or blob that is signed with Azure AD credentials.

Storage permissions – POSIX-like (ACLs) in Azure Data Lake Gen2

Fine grained access to files and directories in a data lake via
Access Control Lists

- Read
- Write
- Execute

Operation	/	Oregon/	Portland/	Data.txt
Read Data.txt	--X	--X	--X	R--
Append to Data.txt	--X	--X	--X	RW-
Delete Data.txt	--X	--X	-lX	---
Create Data.txt	--X	--X	-lX	---
List /	R-X	---	---	---
List /Oregon/	--X	R-X	---	---
List /Oregon/Portland/	--X	--X	R-X	---

Two types of ACLs

- **Access ACLs** control access to an object (files and directories).
- **Default ACLs** are templates for a directory

SAS – two types

- You can create two types of shared access signatures:
- **Service SAS.** The service SAS delegates access to a resource in just one of the storage services: the Blob, Queue, Table, or File service.
- **Account SAS.** The account SAS delegates access to resources in one or more of the storage services. All of the operations available via a service SAS are also available via an account SAS. Additionally, with the account SAS, you can delegate access to operations that apply to a given service, such as **Get/Set Service Properties** and **Get Service Stats**. You can also delegate access to read, write, and delete operations on blob containers, tables, queues, and file shares that are not permitted with a service SAS.

SAS key (shared access signature)

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

Create Account level SAS

Allowed services ⓘ

☒ Blob ☒ File ☒ Queue ☒ Table

Allowed resource types ⓘ

☒ Service ☒ Container ☒ Object

Allowed permissions ⓘ

☒ Read ☒ Write ☒ Delete ☒ List ☒ Add ☒ Create ☒ Update ☒ Process

Start and expiry date/time ⓘ

Start

2018-06-13 10:11:12

End

2018-06-13 18:11:12

(UTC+02:00) --- Current Timezone ---

Allowed IP addresses ⓘ

for example, 168.1.5.65 or 168.1.5.65-168.1.5.70

Allowed protocols ⓘ

☒ HTTPS only ☐ HTTPS and HTTP

Signing key ⓘ

key1

[Generate SAS and connection string](#)

Connection string

BlobEndpoint=https://gaardsted20180613.blob.core.windows.net/QueueEndpoint=https://gaardsted20180613.queue.core.windows.net/FileEndpc

SAS token ⓘ

?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se=2018-06-13T16:11:12Z&st=2018-06-13T08:11:12Z&spr=https&sig=%28Lu132%2BDhh5DiC

Blob service SAS URL

https://gaardsted20180613.blob.core.windows.net/?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se=2018-06-13T16:11:12Z&st=2018-06-13T

File service SAS URL

https://gaardsted20180613.file.core.windows.net/?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se=2018-06-13T16:11:12Z&st=2018-06-13T0

Queue service SAS URL

https://gaardsted20180613.queue.core.windows.net/?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se=2018-06-13T16:11:12Z&st=2018-06-13T

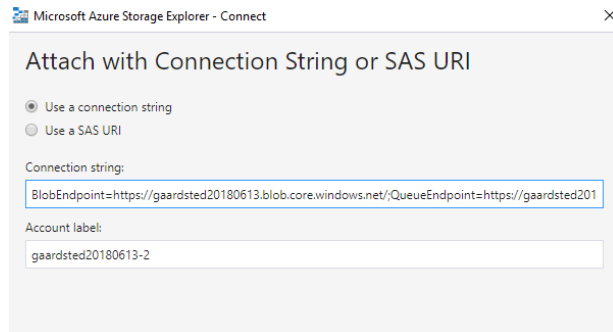
Table service SAS URL

https://gaardsted20180613.table.core.windows.net/?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se=2018-06-13T16:11:12Z&st=2018-06-13T

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. Additionally, any existing SAS tokens will also need to be regenerated. This action will not interrupt access to disks from your virtual machines.

Connect via SAS – Azure Storage Explorer

- Via connection string eller via SAS URI (kan kopieres fra forrige slide)



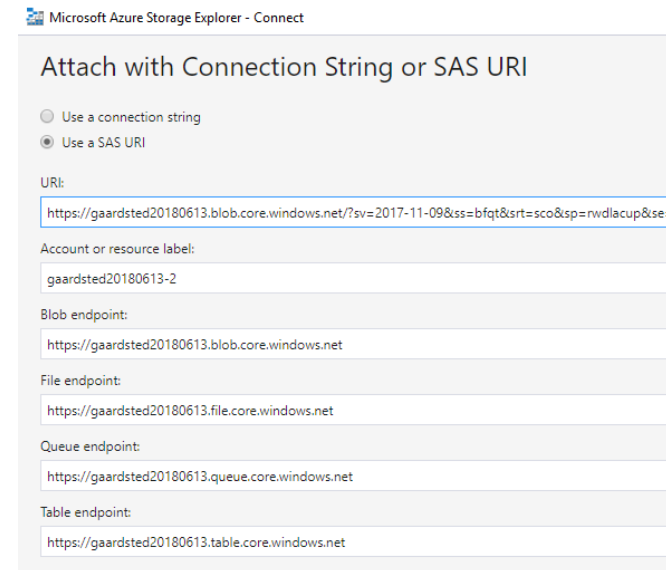
Microsoft Azure Storage Explorer - Connect

Attach with Connection String or SAS URI

☒ Use a connection string
☐ Use a SAS URI

Connection string:
BlobEndpoint=https://gaardsted20180613.blob.core.windows.net/QueueEndpoint=https://gaardsted201

Account label:
gaardsted20180613-2



Microsoft Azure Storage Explorer - Connect

Attach with Connection String or SAS URI

☐ Use a connection string
☒ Use a SAS URI

URI:
https://gaardsted20180613.blob.core.windows.net/?sv=2017-11-09&ss=bfqt&srt=sco&sp=rwdlacup&se

Account or resource label:
gaardsted20180613-2

Blob endpoint:
https://gaardsted20180613.blob.core.windows.net

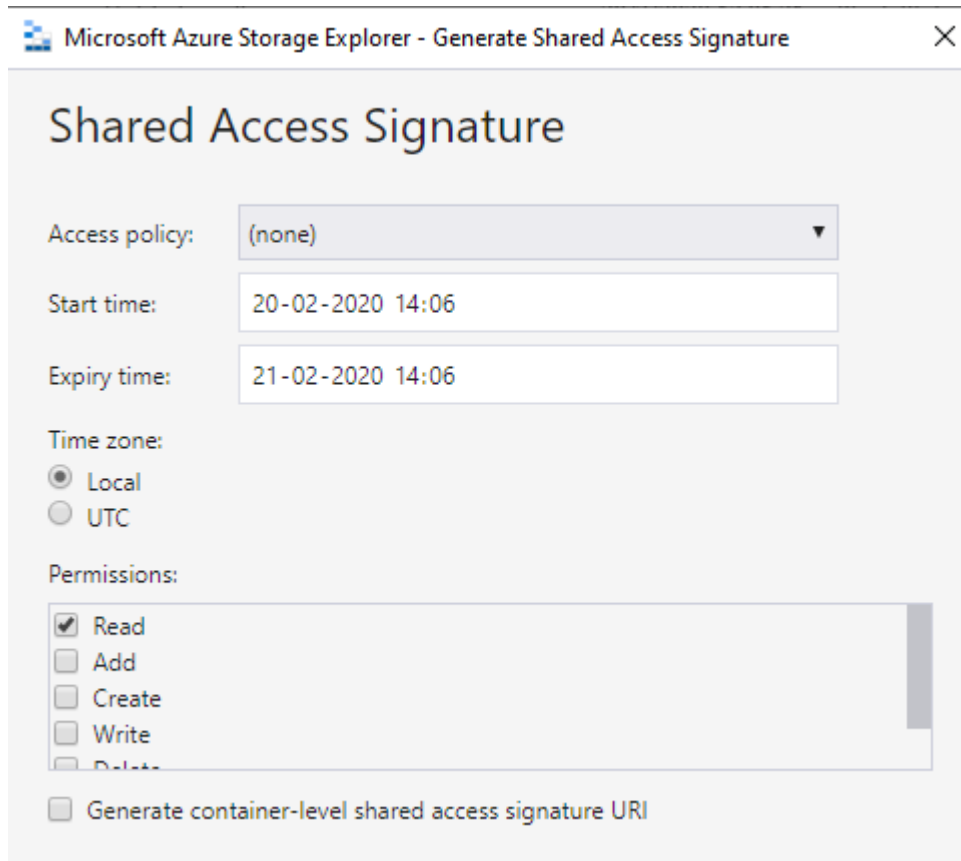
File endpoint:
https://gaardsted20180613.file.core.windows.net

Queue endpoint:
https://gaardsted20180613.queue.core.windows.net

Table endpoint:
https://gaardsted20180613.table.core.windows.net

Create Service Level SAS

- Via Azure Portal, Azure Storage Explorer or PowerShell



The screenshot shows the 'Generate Shared Access Signature' dialog box in Microsoft Azure Storage Explorer. The dialog has a title bar with the application name and a close button. The main content area is titled 'Shared Access Signature'. It contains several input fields and checkboxes:

- Access policy:** A dropdown menu currently showing '(none)'.
- Start time:** A text box containing '20-02-2020 14:06'.
- Expiry time:** A text box containing '21-02-2020 14:06'.
- Time zone:** Two radio buttons, 'Local' (selected) and 'UTC'.
- Permissions:** A list of checkboxes: 'Read' (checked), 'Add', 'Create', 'Write', and 'Delete' (all unchecked).
- Generate container-level shared access signature URI:** An unchecked checkbox at the bottom.

```
New-AzStorageAccountSASToken  
New-AzStorageBlobSASToken  
New-AzStorageContainerSASToken  
New-AzStorageFileSASToken  
New-AzStorageQueueSASToken  
New-AzStorageShareSASToken  
New-AzStorageTableSASToken
```

Storage account – Blob filesystem driver

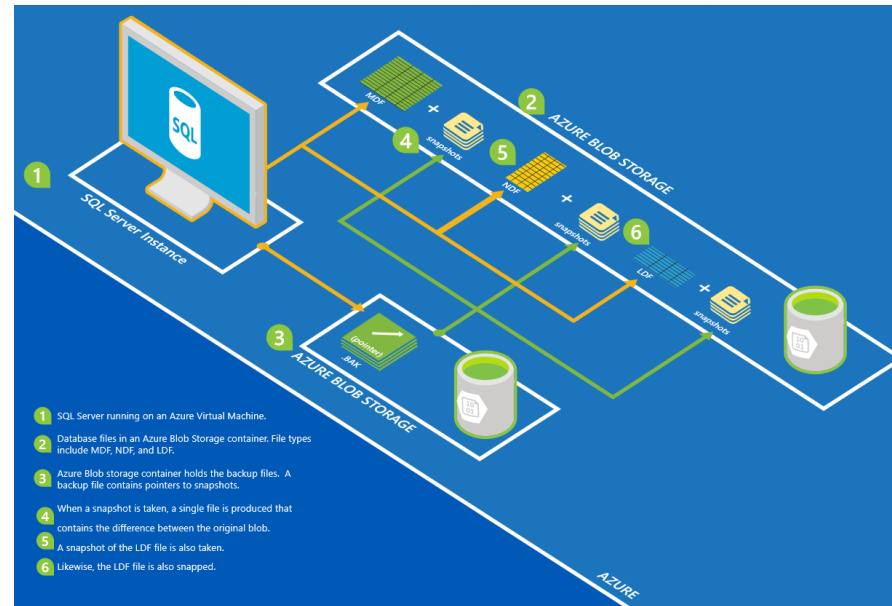
- wasb - Windows Azure Storage Blob driver
 - Authenticate via
- Abfs – Azure Blob Filesystem Driver
 - Authentication via
 - Shared Key
 - Azure AD

SQL Server: Azure storage snapshot backup

Azure storage snapshot backup

SQL Server File-snapshot backup uses Azure snapshots to provide nearly instantaneous backups and quicker restores for database files stored using the Azure Blob storage service.

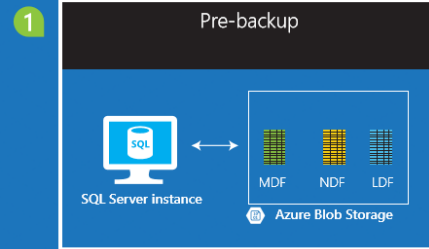
A file-snapshot backup consists of a set of Azure snapshots of the blobs containing the database files plus a backup file containing pointers to these file-snapshots. Each file-snapshot is stored in the container with the base blob. You can specify that the backup file itself to be written to URL, disk or tape. Backup to URL is recommended.



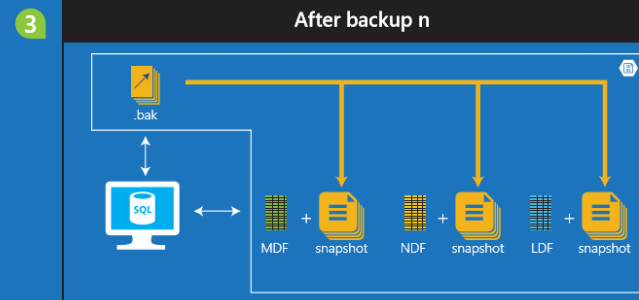
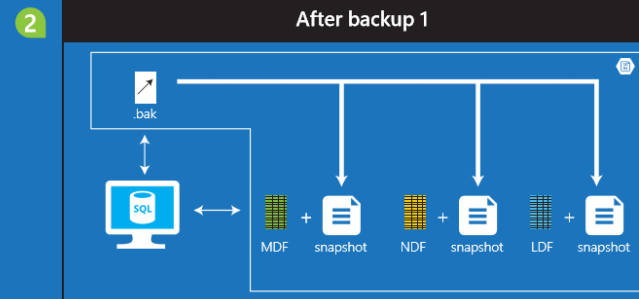
Backup to Azure Blob Storage

Backup to Azure Blob Storage

Backup to Azure with file snapshots is a feature of SQL Server 2016.



- 1 SQL Server instance with Data and Log files on Azure Blob Storage.
- 2 Snapshot backup taken: 1 .bak file points to actual snapshots associated with original blob stored in Azure Blob storage.
- 3 Multiple .bak files. Each points to its own set of snapshots for the backup.



SQL Server datafiles in Azure

- A SAS token can be generated via
 - Azure portal
 - Azure Storage Explorer
 - PowerShell
 - SQL Server Management Studio (SSMS)

The SQL Server needs the SAS token to connect to the storage container – therefore we create a CREDENTIAL with the container url and the SAS token

Snapshot backup

- **Full database backup:** Performing a full database backup using file-snapshot backup creates an Azure snapshot of each data and log file comprising the database, establishes the transaction log backup chain, and writes the location of the file-snapshots into the backup file.
- **Transaction log backup:** Performing a transaction log backup using file-snapshot backup creates a file-snapshot of **each database file (not just the transaction log)**, records the file-snapshot location information into the backup file, and **truncates the transaction log** file.
- After the initial full backup that is required to establish the transaction log backup chain (which can be a file-snapshot backup), **you only need to perform transaction log backups** because each transaction log file-snapshot backup set contains file-snapshots of all database files and can be used to perform a database restore or a log restore. After the initial full database backup, you do not need additional full or differential backups because the Azure Blob storage service handles the differences between each file-snapshot and the current state of the base blob for each database file.

Backup to Azure blob storage - syntax

- Initial full backup

```
BACKUP DATABASE AdvCloud  
TO URL = 'https://gaardsted.blob.core.windows.net/mssql/AdvCloud.bak'  
WITH FILE_SNAPSHOT;
```

- And then afterwards just make transaction log backups:

```
BACKUP LOG AdvCloud  
TO URL = 'https://gaardsted.blob.core.windows.net/mssql/AdvCloud.trn'  
WITH FILE_SNAPSHOT;
```

- `SELECT * FROM sys.fn_db_backup_file_snapshots ('AdvCloud');`

Restore database from file snapshot

- We can just use a single transaction log backup snapshot(!):

```
RESTORE DATABASE AdventureWorks2016
```

```
FROM URL = 'https://gaardsted.blob.core.windows.net/mssql/AdventureWorks2016_Log.trn'
```

```
WITH RECOVERY, REPLACE;
```

- Create a new database from a transaction log snapshot backup:

```
RESTORE DATABASE [AdvCloud2] FROM URL = 'https://gaardsted.blob.core.windows.net/mssql/AdventureWorks.trn' WITH FILE = 1
```

```
, MOVE 'AdvCloud' TO N'https://gaardsted.blob.core.windows.net/mssql/AdvCloud2.mdf'
```

```
, MOVE 'AdvCloud_log' TO N'https://gaardsted.blob.core.windows.net/mssql/AdvCloud2_log.ldf'
```

File snapshot backups for database files i Azure

- The maximum number of snapshots that you can store is 100.
- The frequency of backups can be no shorter than 10 minutes
- When using file-snapshot backups, you cannot perform an Online Restore

- Deleting the base blob will invalidate the backup set and you are prevented from dropping a blob that contains file-snapshots (unless you expressly choose to delete a blob with all of its file-snapshots). Furthermore, dropping a database or a data file does not delete the base blob or any of its file-snapshots. Also, deleting the backup file does not delete any of the file-snapshots in the backup set. To delete a file-snapshot backup set, use the **sys.sp_delete_backup** system stored procedure.