Microsoft Azure Storage

Enabling the
Digital Enterprise

MICROSOFT AZURE STORAGE (BLOB/TABLE/QUEUE)
July 2015

The goal of this white paper is to explore Microsoft Azure Storage, understand how it works and to get hands-on experience working with it.

The goal of Microsoft Azure Storage is to allow users and applications to:

- Access data efficiently from anywhere at anytime
- Store data for any length of time
- Scale to store any amount of data
- Be confident that the data will not be lost
- Pay for only what used/stored

Sagar Bandal, Avdhoota Narayandasani, Snehal Khandkar, Shailaja Patole
Arrk Group

AZURE BLOB STORAGE

WHAT IS BLOB STORAGE?

Azure Blob storage is a service for storing large amounts of unstructured data, such as text or binary data, that can be accessed from anywhere in the world via HTTP or HTTPS. You can use Blob storage to expose data publicly to the world, or to store application data privately.

Common uses of Blob storage include:

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

BLOB SERVICE CONCEPTS

The Blob service contains the following components:

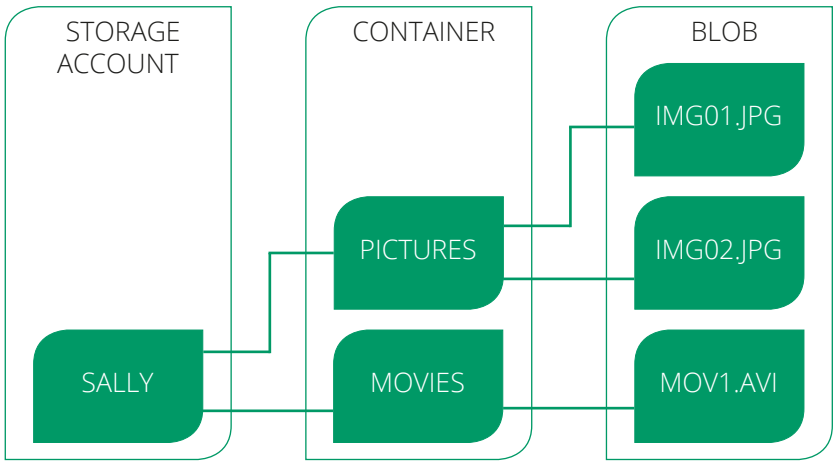

FIGURE 1

Storage Account | All access to Azure Storage is done through a storage account.

Container | A container provides a grouping of a set of blobs. All blobs must be in a container. An account can

contain an unlimited number of containers and a container can store an unlimited number of blobs.

Blob | A file of any type and size. There are two types of blobs that can be stored in Azure Storage: block and page blobs. Most files are block blobs. A single block blob can be up to 200 GB in size. This tutorial uses block blobs. Page blobs, another blob type, can be up to 1 TB in size, and are more efficient when ranges of bytes in a file are modified frequently.

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

http://<storage account>.blob.core.windows.net/<container>/<blob>

The following example URL could be used to address one of the blobs in the diagram above:

http://sally.blob.core.windows.net/movies/MOV1.AVI

BLOB FEATURES & FUNCTIONS

- Store Large Objects (up to 50 GB each)
- Standard REST PUT/GET Interface

  http://<Account>.blob.core.windows.net/<Container>/<BlobName>

  | PutBlob - Inserts a new blob or overwrites the existing blob
  | GetBlob - Get whole blob or by starting offset, length
  | DeleteBlob
  | Support for Continuation on Upload
- Associate Metadata with Blob
  Metadata is <name, value> pairs
  Set/Get with or separate from blob data bits, Up to 8KB per blob

BLOB REST INTERFACE

All access to Microsoft Azure Blob is done through a standard HTTP REST PUT/GET/DELETE interface.

The HTTP/REST commands supported to implement the blob operations include:

- PUT Blob | Insert a new blob or overwrite an existing blob of the given name
- GET Blob | Get an entire blob, or get a range of bytes within the blob using the standard HTTP range GET operation
- DELETE Blob | Delete an existing blob
- CopyBlob | Copy a blob from a source blob to a destination blob within the same storage account. This copies the whole committed blob, including the blob metadata, properties, and committed blocklist. You can use CopyBlob along with DeleteBlob to rename a blob, to move a blob between containers, and to create backup copies of your existing blobs
- Get Block List | Retrieve the list of blocks that have been uploaded as part of a blob. There are two block lists maintained for a blob, and this function allows retrieval of either of the two or both:

  Committed Block List | This is the list of blocks that have been successfully committed as part of a PutBlockList for a given blob

  Uncommitted Block List | This is the list of blocks that have been uploaded for a blob since the last PutBlockList for the blob. These blocks represent the temporary/uncommitted blocks that have not yet been committed

All of these operations can be done on a blob with the following URL:

http://<account>.blob.core.windows.net/<container>/<blobname>

You can upload a blob up to 64MB in size using a single PUT blob request up into the cloud. To go up to the 50GB blob size limit, one must use the block interface.

A BLOB AS A LIST OF BLOCKS

One of the target scenarios for Microsoft Azure Blob is to enable efficient upload of blobs that are many GBs in size. This is provided by Microsoft Azure Blob through the following steps:

- Break the Blob (e.g. Movie.avi) to be uploaded into contiguous blocks. For example, a 10GB movie can be broken up into 2500 blocks, each of size 4MB, where the first block represents bytes 1 through 4194304, the second block would be bytes 4194305 through 8388608, etc
- Give each block a unique ID. This unique ID is scoped by the blob name being uploaded. For example, the first block could be called “Block 0001”, the second block “Block 0002”, etc
- PUT each block into the cloud. This is done by doing a PUT specifying the URL above with the query specifying that this is a PUT block along with the block ID. In continuing our example, to put the first block, the blob name would be “Movie.avi”, and the block ID is “Block 0001”
- After all of the blocks are stored in Microsoft Azure Storage, then we commit the list of uncommitted blocks uploaded to represent the blob name they were associated with. This is done with a PUT specifying the URL above with the query specifying that this is a blocklist command. Then the HTTP header contains the list of blocks to be committed for this blob. When this operation succeeds, the list of blocks, in the order in which they were listed, now represents the readable version of the blob. The blob can then be read using the GET blob commands described above

The following figure incorporates blocks into the Microsoft Azure Blob data concepts.

As described earlier, blobs can be accessed via PUT and GET by using the following URL:

http://<account>.blob.core.windows.net/<container>/<blobname>

In the examples shown in Figure 2, a single PUT can be used to put the images with the following URLs:

http://sally.blob.core.windows.net/pictures/IMG001.JPG
http://sally.blob.core.windows.net/pictures/IMG002.JPG

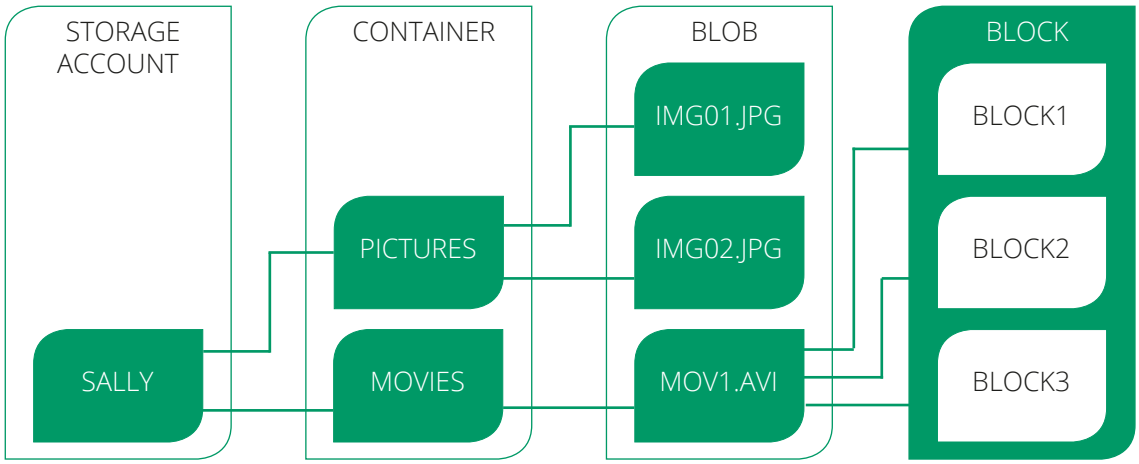

FIGURE 2

The same URLs can be used to get the blobs. In using a single PUT, blobs up to 64MB can be stored. To store blobs larger than 64MB and up to 50GB, one needs to first PUT all of the blocks, and then PUT the block list to comprise the readable version of the blob. In Figure 2 above, only after the blocks have been PUT and committed as part of the block list can the blob be read using the following URL:

http://sally.blob.core.windows.net/pictures/MOV1.AVI

GET operations always operate on the blob level, and do not involve specifying blocks.

AZURE TABLE STORAGE

WHAT IS THE TABLE SERVICE?

The Azure Table storage service stores large amounts of structured data. The service is a NoSQL data store which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of the Table service 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 the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

TABLE SERVICE CONCEPTS

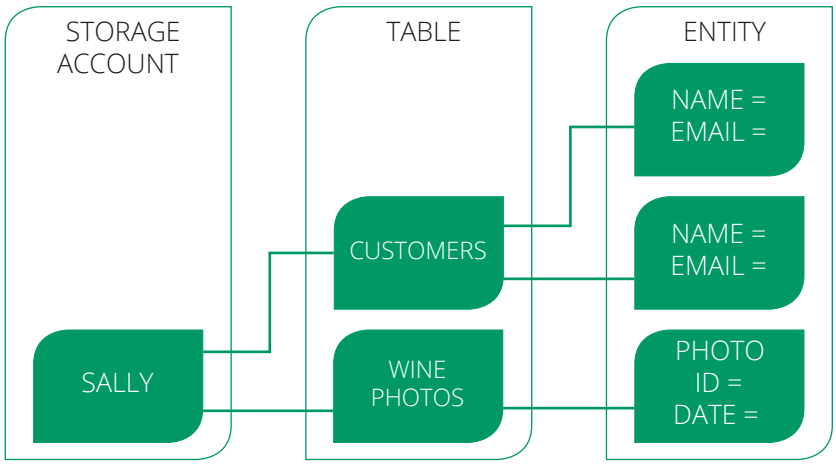

FIGURE 3

The Table service contains the following components:

- URL format | Code addresses tables in an account using this address format:

  http://<storage account>.table.core.windows.net/<table>

  You can address Azure tables directly using this address with the OData protocol.

- Storage Account | All access to Azure Storage is done through a storage account. See Azure Storage Scalability and Performance Targets for details about storage account capacity.

- Table | A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. The number of tables that a storage account can contain is limited only by the storage

account capacity limit

- Entity | An entity is a set of properties, similar to a database row. An entity can be up to 1MB in size

- Properties | A property is a name-value pair. Each entity can include up to 252 properties to store data. Each entity also has 3 system properties that specify a partition key, a row key, and a timestamp. Entities with the same partition key can be queried more quickly, and inserted/updated in atomic operations. An entity's row key is its unique identifier within a partition

PARTITION KEY AND PARTITIONS

- Every Table has a Partition Key | It is the first property (column) of your Table. Used to group entities in the Table into partitions

- A Table Partition | All entities in a Table with the same partition key value

- Partition Key is exposed in the programming model | Allows application to control the granularity of the partitions and enable scalability

PURPOSE OF THE PARTITION

- Performance and Entity Locality | Entities in the same partition will be stored together. Efficient querying and cache locality

- Table Scalability | We monitor the usage patterns of partitions. Automatically load balance partitions. Each partition can be served by a different storage node. Scale to meet the traffic needs of your table

ROWKEY IN TABLE STORAGE

A RowKey in Table Storage is a very simple thing: it's your 'primary key' within a partition. PartitionKey + RowKey form the composite unique identifier for an entity. Within one

PartitionKey, you can only have unique RowKey. If you use multiple partitions, the same RowKey can be reused in every partition. Therefore, a RowKey is just the identifier of an entity within a partition.

TABLE OPERATIONS

- Create a table
- Add an entity to a table
- Insert a batch of entities
- Retrieve all entities in a partition
- Retrieve a range of entities in a partition
- Retrieve a single entity
- Replace an entity
- Insert or replace an entity
- Query a subset of entity properties
- Delete an entity
- Delete a table
- Retrieve entities in pages asynchronously

TRANSACTIONS

One of the most common questions that come up when talking about Table Storage is regarding whether transactions are supported. Table storage does support batch transactions against data within the same table and the same partition.

There are several rules about the transactions though: all of the entities must exist within the same partition of the table, the number of entities in the transaction can't exceed 100 and the entire batch being sent to the server cannot exceed 4 MB in size.

As you can see, there are limitations to the level of transaction support you get which revolves around the partition. This is another good reason why choosing your partition key scheme is very important.

TABLE STORAGE v SQL DATABASE

Similar to Azure SQL Database, Azure Table Storage stores structured data, with the main difference being that Azure SQL Database is a relational database management system based on the SQL Server engine and built on standard relational principles and practices. As such, it provides relational data management capabilities through Transact-SQL queries, ACID transactions and stored procedures that are executed on the server side.

Azure Table Storage is a flexible key/value store that enables you to build cloud applications easily, without having to lock down the application data model to a particular set of schemas. It is not a relational data store and does not provide the same relational data management functions as Azure SQL Database (such as joins and stored procedures).

CAMPARISON CRITERIA	AZURE TABLE STORAGE	AZURE SQL DATABASE
Data Relationships	No	Yes
Server-side Processing	No	Yes
Transaction Support	Limited	Yes
Geo-replication	Yes	No
Table Schema	Relaxed	Managed
Similarity [to existing data stores used on-premises]	No	Yes
Scale-out	Automatic	Manual
Data Types	Simple	Simple, Complex and User Defined

TABLE 1

Azure Table Storage provides limited support for server-side queries, but does offer transaction capabilities. Additionally, different rows within the same table can have different structures in Azure Table Storage. This schema-less property of Azure Tables also enables you to store and retrieve simple relational data efficiently.

If your application stores and retrieves large data sets that do not require rich relational capabilities, Azure Table Storage might be a better choice. If your application requires data processing over schematised data sets and is relational in nature, Azure SQL Database might better suit your needs. There are several other factors you should

consider before deciding between Azure SQL Database and Azure Table Storage. Some of these considerations are listed in the Table 1 on page 12.

AZURE QUEUE STORAGE

WHAT IS QUEUE STORAGE?

Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size and a queue can contain millions of messages, up to the total capacity limit of a storage account. A storage account can contain up to 500 TB of blob, queue and table data.

See Azure Storage Scalability and Performance Targets for details about storage account capacity.

Common uses of queue storage include:

- Creating a backlog of work to process asynchronously
- Passing messages from an Azure Web role to an Azure Worker role

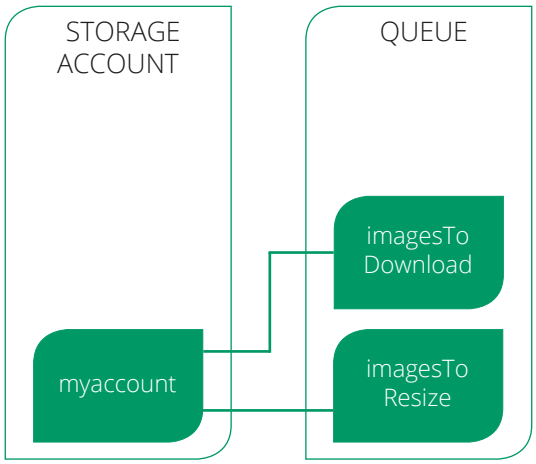

FIGURE 4

QUEUE SERVICE CONCEPTS

The Queue service contains the following components:

- URL format | Queues are addressable using the following URL format:

  http://<storage account>.queue.core.windows.net/<queue>

  The following URL addresses one of the queues in the diagram.

  http://myaccount.queue.core.windows.net/imagesToDownload

- Storage Account | All access to Azure Storage is done through a storage account
- Queue | A queue contains a set of messages. All

messages must be in a queue
- Message | A message, in any format, of up to 64KB

FUNDAMENTAL DATA ABSTRACTIONS – QUEUES

Provide reliable storage and delivery of messages for an application.

- The decision on when to use Microsoft Azure Queues or Service Bus Queues depends on a number of factors, dependent heavily on the individual needs of the application and its architecture. If an application already uses the core capabilities of Microsoft Azure, you may prefer to choose Microsoft Azure Queues, especially if you require basic communication and messaging between services or need queues that can be larger than 5 GB in size

- When a message is read from the queue, the consumer is expected to process the message and then delete it. After the message is read, it is made invisible to other consumers for a specified interval. If the message has not yet been deleted at the time the interval expires, its visibility is restored, so that another consumer may process it

MICROSOFT AZURE QUEUES

- Provide reliable message delivery
- Simple, asynchronous work dispatch
- Programming semantics ensure that a message can be processed at least once
- Queues are Highly Available, Durable and Performance Efficient
- Provide reliable message delivery
- Allows Messages to be retrieved and processed at least once
- No limit on number of messages stored in a Queue
- Message size is <=64KB
- Access is provided via REST

ACCOUNT, QUEUES AND MESSAGES

- An Account can create many Queues
- Queue Name is scoped by the Account
- A Queue contains Messages
- No limit on number of messages stored in a Queue. But a Message is stored for at most a week.

http://<Account>.queue.core.windows.net/<QueueName>

- Message Size <= 64 KB. To store larger data, store data in blob/entity storage, and the blob/entity name in the message

QUEUE PROGRAMMING API

- Queues | Create/Clear/Delete Queues. Inspect Queue Length
- Messages | Enqueue (QueueName, Message). Dequeue (QueueName, Invisibility Time T). Delete (QueueName, MessageID)

QUEUE SERVICE CONCEPTS

- Addressing Queue Service Resources | The Queue service exposes the following resources via the REST API :

  Account | The storage account is a uniquely identified entity within the storage system. The account is the parent namespace for the Queue service. All queues are associated with an account.

  Queue | A queue stores messages that may be retrieved by a client application or service.

  Messages | Messages are UTF-8 encoded text that can be the value of an XML element. A message can be 64 KB in size.

- Naming Queues and Metadata

- Settings Timeouts for Queue Service Operations

Get In Touch

+44 (0) 161 227 9900
www.arrkgroup.com
talktous@arrkgroup.com
@arrkgroup
UK:
Greenheys, Pencroft Way,
Manchester Science Park,
Manchester, M15 6JJ
India:
Building 5, Sector 2,
Millennium Business Park,
Mahape, Navi Mumbai - 400 710

About Arrk Group

Arrk Group is a software engineering company with core competency in three areas:

- Building easy-to-use, robust digital applications and platforms
- User centred design and high performing engineering practices
- Through dedicated, distributed customer teams

Founded in 1998 we have designed and delivered award-winning large scale bespoke digital engagement platforms and membership management systems and have a strong track record of delivering digital transformation in a variety of industries.

Our approach brings together our experience in user centred design with leading Agile based software engineering practices, delivered by distributed teams to provide intuitive but industrial strength solutions that focus on business outcomes. As we provide digital applications and platforms, our solutions have to be engaging and easy to use.

We prefer to automate much of the testing, continuous integration and build and deployment pipelines. Our development skillsets encompass a broad range of technologies including the Microsoft technology stack, .NET, Java and J2EE, Rails, Grails, Cloud based solutions and Mobile Development.

Our SpArrk communities programme challenges our consultants to stay at the leading edge of technologies and processes and follow key trends in our industry to generate thought leadership and internal R&D. The SpArrk communities focus on 3 main areas:

- Digital Platform & Architecture
- High Performance Software Engineering
- Leading Edge Technology