

A decorative graphic on the left side of the slide consisting of orange lines and circles, resembling a circuit board or a stylized tree structure, extending from the top to the bottom.

AZURE STORAGE

MODULE: APPLICATION DEVELOPMENT 3A

CONTENT DEVELOPED BY: CASSIM VANKER

LEARNING OBJECTIVES

- Azure Storage
 - Azure Storage Concepts
 - Blobs
 - Block Blobs
 - Table Blobs
 - Tables
 - Partitionkey
 - RowKey
 - Deciding on a partitionkey and rowkey
 - Storage Queues
 - Drive
 - Storage Architecture
 - Know The Scalability Targets
 - Geo-replication
 - Concurrency optimistic – Blobs
 - Concurrency pessimistic – Blobs
 - Concurrency optimistic – Tables
 - Concurrency - Queues
- Demo

AZURE STORAGE

Overview

Blobs

File system in the cloud

Tables

Scalable structured storage

Queues

Reliable storage and delivery of messages

Drives

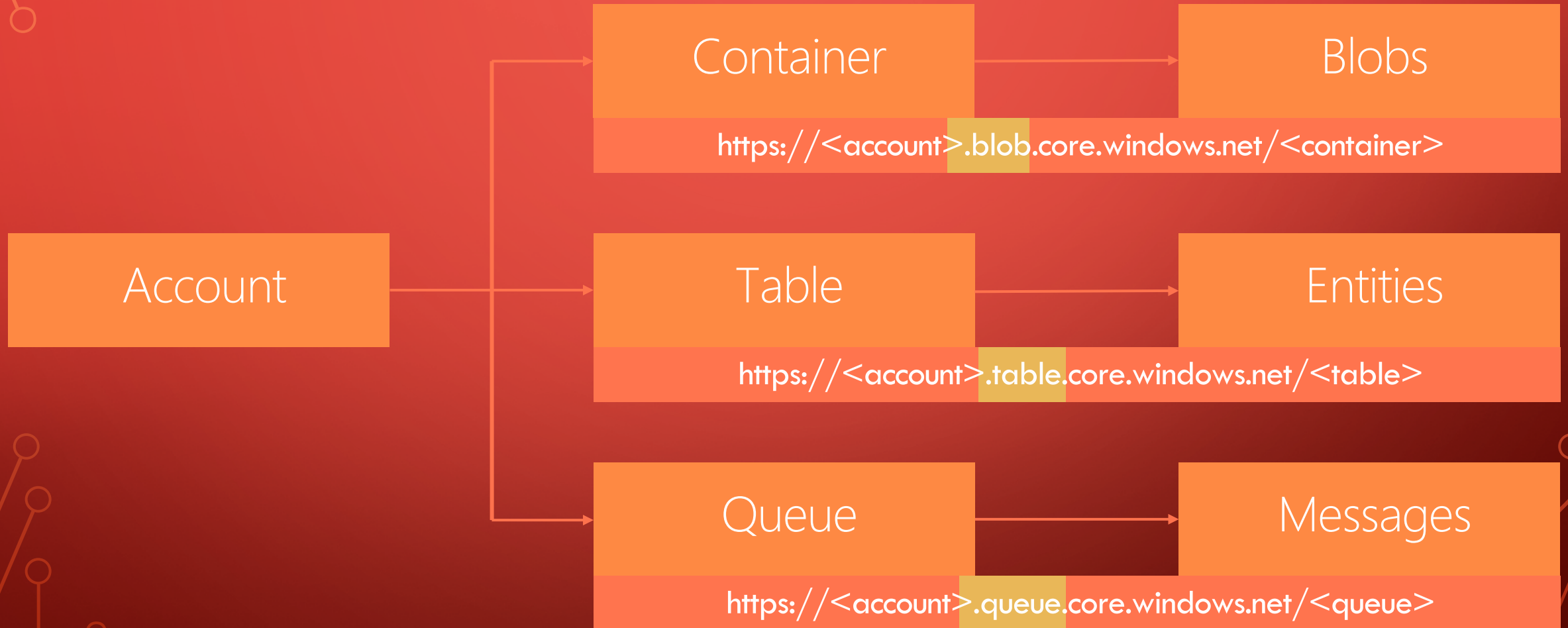
NTFS volumes for Azure applications

Client access Via:

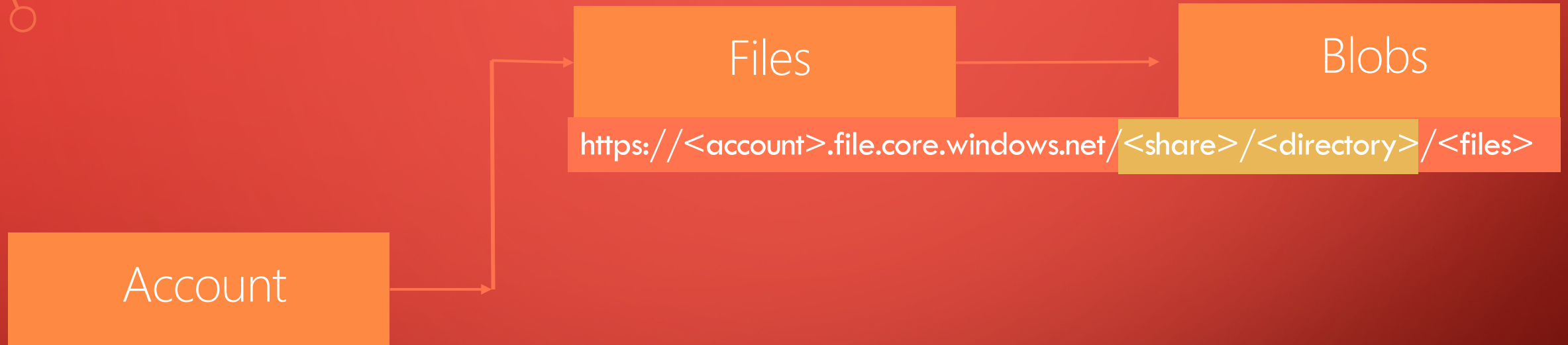
REST APIs and Client Libraries

NTFS APIs for Windows Azure Drives

AZURE STORAGE CONCEPTS



AZURE STORAGE CONCEPTS



FILES

- Supports the SMB 2.1 /3.0 protocol.
- The Server Message Block (SMB) Protocol is a network file sharing protocol, and as implemented in Microsoft Windows is known as Microsoft SMB Protocol.
- SMB communicates over TCP port 445. The port needs to be opened.
- An account can contain an unlimited number of shares, and a share can store an unlimited number of **files**, up to the 5 TB total capacity of the **file** share.
- Azure File share can be mounted in File Explorer

BLOBS

- A highly scalable and durable file system in the cloud.
- Store files as blobs and associate metadata with it
- Suitable for storing large amounts of unstructured data
- Blob names are case-sensitive.
- Blobs can be up to 200 GB in size
- Provides continuation for large uploads
- Provides range reads
- Strong Consistency (always return the same value) and Optimistic Concurrency (verify if the data has change since the last read).
- Exclusive write lease
- Create versions/backup/snapshot's of your blobs
- Every blob request must be signed with the account owner's key
- Three types of blobs Block, Append Blobs and Page blobs
- Share your files
 - The container must be public
 - Shared Access Signature (SAS) – share pre-authenticated URLs with users

BLOCK BLOBS

- Upload large blobs efficiently
- Block blobs are comprised of blocks, each of which is identified by a block ID.
- Create or modify a block blob by writing a set of blocks and committing them by their block IDs.
- The maximum size of a block blob is therefore slightly more than 4.75 TB
- With a block blob, you can upload multiple blocks in parallel to decrease upload time.
- Each block can include an MD5 hash to verify the transfer, so you can track upload progress and re-send blocks as needed.

PAGE BLOBS

- Page blobs are a collection of 512-byte pages optimised for random read and write operations.
- To create a page blob, you initialize the page blob and specify the maximum size the page blob will grow.
- A write to a page blob can overwrite just one page, some pages, or up to 4 MB of the page blob.
- Writes to page blobs are immediate and in-place.
- The maximum size for a page blob is 8 TB.
- Page blobs are optimized for high-speed random access read and write operations.
- Page blobs are perfect for storing virtual file systems (like VHDs).

APPEND BLOBS

- An append blob is comprised of blocks and is optimized for append operations.
- When you modify an append blob, blocks are added to the end of the blob only.
- Updating or deleting of existing blocks is not supported.
- Each block in an append blob can be a different size, up to a maximum of 4 MB.
- The maximum size of an append blob is slightly more than 195 GB.

TABLES

- Scalable Structured Storage
- Key/Attribute store
- Tables can store billions of entities and TBs of data
- Provides flexible schema (NoSQL)
- Azure Tables only support 255 properties (columns) on a single entity (row) and a max row size of 1MB.
- Single columns are limited to 64 KB of binary data.
- Has A Data Model
- A table is a set of entities (rows)
- An entity is a set of properties (columns)
- Tables consist of Partitonkey and Rowkey properties
- Familiar and Easy to use API
- OData Protocol
- WCF Data Services - .NET classes and LINQ

TABLES — PARTITIONKEY

- *A table's entities are organized by partition.*
- *A partition is a consecutive range of entities possessing the same partition key value.*
- *The partitionkey is a unique identifier for the partition within a given table, specified by the PartitionKey property.*
- *The partition key forms the first part of an entity's primary key.*
- *The partition key may be a string value up to 32 KB in size.*
- *You must include the PartitionKey property in every insert, update, and delete operation.*

TABLES – ROWKEY

- The second part of the primary key is the row key.
- The row key is a unique identifier for an entity within a given partition.
Together the PartitionKey and RowKey uniquely identify every entity within a table.
- The row key is a string value that may be up to 32 KB in size.
- You must include the RowKey property in every insert, update, and delete operation.

DECIDING ON A PARTITIONKEY AND ROWKEY

- Use and Example of a Customer, Order and Orderline.
- Customer (PK: sales region, RK: customer id) – it enables fast searches on region and on customer id
- Order (PK: customer id, RK; order id) – it allows me to quickly fetch all orders for a specific customer (as they are colocated in one partition), it still allows fast querying on a specific order id as well)
- Orderline (PK: order id, RK: order line id) – allows fast querying on both order id as well as order line id.

DECIDING ON A PARTITIONKEY AND ROWKEY

- Customer (PK: customer id, RK: display name) – it enables fast searches on customer id and display name
- Order (PK: customer id, RK; order id) – it allows me to quickly fetch all orders for a specific customer (as they are colocated in one partition), it still allows fast querying on a specific order id as well)
- Orderline (PK: order id, RK: item id) – allows fast querying on both order id as well as the item bought, of course given that one order can only contain one order line for a specific item (PK + RK should be unique).

STORAGE QUEUES

- Features a simple REST-based GET/PUT/PEEK interface, providing reliable, persistent messaging within and between services.
- Messages in Storage queues are typically first-in-first-out, but sometimes they can be out of order
- Users can Put message into the queue
- A storage account can create any number of queues
- Get message makes the message invisible in queue for a specified invisibility timeout (default 30 seconds)
- User must Delete message once done processing to remove message from queue
- If worker crashes, message becomes visible for another worker to process
- Max queue size is **500 TB**
- Max message size 64KB and default expiry of 7 days
- Any number of concurrent clients can access the queue.
- Programming semantics – Ensures that a message can be processed at least once

STORAGE QUEUES

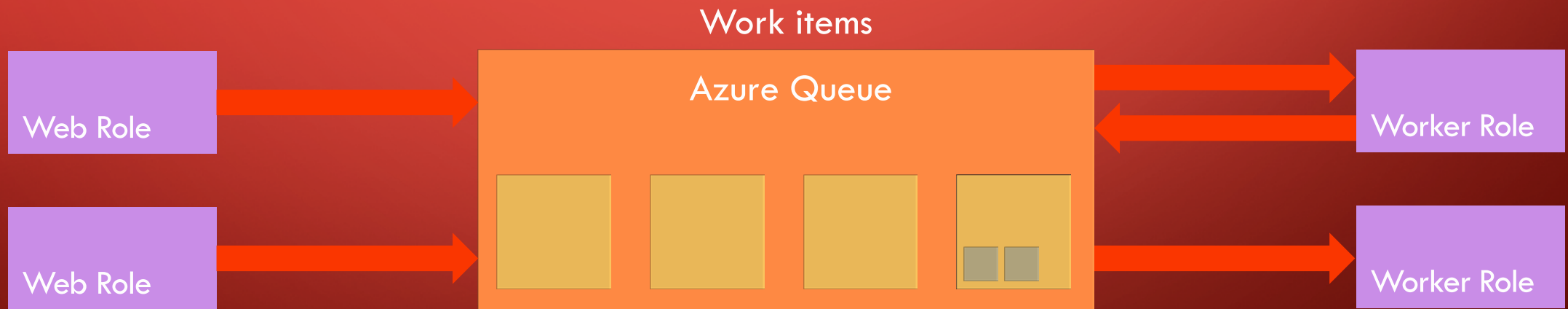
- Use message count to dynamically increase/reduce workers.
 - Retain one instance that polls once every X time period
 - One instance polling every second results in 2,678,400 calls per month
 - Costs around \$2.67
 - Spawn more instances when you detect backlog
- Steps to processing messages
 - A producer put's a message into the queue
 - A consumer takes the message off the queue.
 - The message becomes invisible for a time out period
 - Once the consumer is done, the message is deleted from the queue.
 - Should an error occur then after the time out period the message is returned to the queue.

QUEUES

Periodically store progress information in message content

Extend visibility timeout with another 5 minutes

Get Message with 5 minutes visibility timeout



DRIVE

- A durable NTFS volume for Windows Azure applications
 - Uses NTFS APIs
 - Easy migration path to the cloud
 - Durability and survival of data on application failover or hardware failure.
- Drives can be up to 1 TB
- Uses SMB 3.0 protocol

KNOW THE SCALABILITY TARGETS

Single Blob Partition

- Throughput up to 60 MB/s

Single Queue/Table Partition

- Up to 500 transactions (entities or messages) per second

Storage Account

- SLA – 99.9% Availability
- Capacity – Up to 100 TBs
- Transactions – Up to 5000 entities per second
- Bandwidth – Up to 3 gigabits per second

GEO-REPLICATION

By default enabled

Provide data durability in face of major data center disasters

Data only geo-replicated within regions

The other location in region is the secondary location

Off critical path of live requests

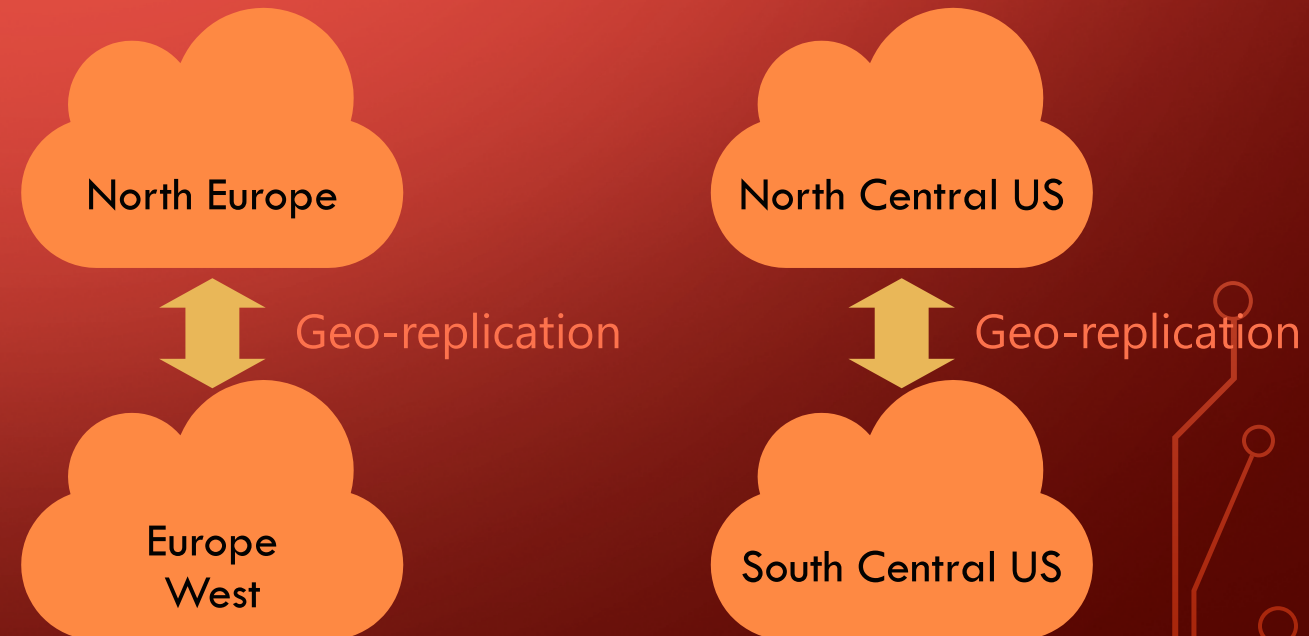
Included in the price of the storage account



Data geo-replicated cross data centers 100s miles apart

User chooses primary location during account creation

Asynchronous geo-replication



CONCURRENCY OPTIMISTIC - BLOBS

- Every object is assigned an identifier.
- This identifier is updated when an update is performed on the object.
- The identifier is returned to the client as an ETag.
- When a user performs an update on an object. The user can ask Azure to check the original ETag to ensure that the Etags match.
- This is called optimistic concurrency.

CONCURRENCY PESSIMISTIC - BLOBS


- To lock a blob for exclusive use you need to acquire a lease on it.
- When you acquire a lease, you specify the lease period.
- A lease can last between 15 to 60 seconds or infinite (exclusive lock).
- A finite lease can be renewed to extend it, and you can release any lease when you are finished with it.
- The blob service automatically releases finite leases when they expire.
- The storage service enforces exclusive writes (put, set and delete operations) on leases.
- In order to ensure exclusivity for read operations developer need to ensure that all client applications use a lease ID. Furthermore, the developer must ensure that only one client at a time has a valid lease ID.

CONCURRENCY OPTIMISTIC - TABLES

- Optimistic Concurrency is automatically catered for using Etags.
- Developers should rely on optimistic concurrency when developing scalable applications.
- Concurrency is performed on the following operations: Update Entity, Merge Entity and Delete Entity.

CONCURRENCY - QUEUES

- When a message is retrieved from the queue, the response includes the message and a pop receipt value, which is required to delete the message.
- The message is not automatically deleted from the queue, but it is not visible to other clients for the time interval specified by the `visibilitytimeout` parameter.
- The client that retrieves the message is expected to delete the message after it has been processed and before the time specified by the `TimeNextVisible` element of the response.

A decorative graphic on the left side of the slide, consisting of a network of thin, light-orange lines that resemble a circuit board or a neural network. These lines are connected to small, hollow orange circles at various points, creating a complex, branching pattern that extends from the top to the bottom of the frame.

DEMO'S – CREATING BLOBS, TABLES, QUEUES ON AZURE PORTAL