



# Windows and SQL Azure Storage Deep Dive

Rainer Stropek  
software architects gmbh  
[rainer@software-architects.at](mailto:rainer@software-architects.at)

# Abstract

Elasticity is what the world of cloud computing is all about. Do you quickly need more storage because of high load on your systems? No problem, in Windows Azure you can store up to 100 TB with a single account and create new SQL clusters within a few seconds. And the best: you just pay for what you really use. In this one day workshop Rainer Stropek, MVP for Windows Azure, presents the storage technologies of Windows and SQL Azure. Learn about blob and table storage as well as SQL Azure, Microsoft's SQL Server in the cloud. Rainer will start by comparing the different storage options and giving you advice when to use what. Next you will see all storage systems of the Windows Azure Platform in action. The third part of the workshop dives deeper into SQL Azure. Rainer will cover the underlying architecture of SQL Azure including its security and firewall capabilities. You will see the differences between on-premise SQL Server and SQL Azure and explore performance and cost benefits of the different sharding approaches that are typical for scale-out scenarios in the cloud.

# Introduction

- software architects gmbh
- Rainer Stropek

Developer, Speaker, Trainer

MVP for Windows Azure

[rainer@timecockpit.com](mailto:rainer@timecockpit.com)



@rstropek



<http://www.timecockpit.com>  
<http://www.software-architects.com>

IT &  
Dev CONNECTIONS  
powered by Microsoft®

# Agenda

- Storage in the Windows Azure Platform
- Windows Azure Storage
  - Blob Storage
  - Table Storage
  - Queues
  - Drives
- SQL Azure

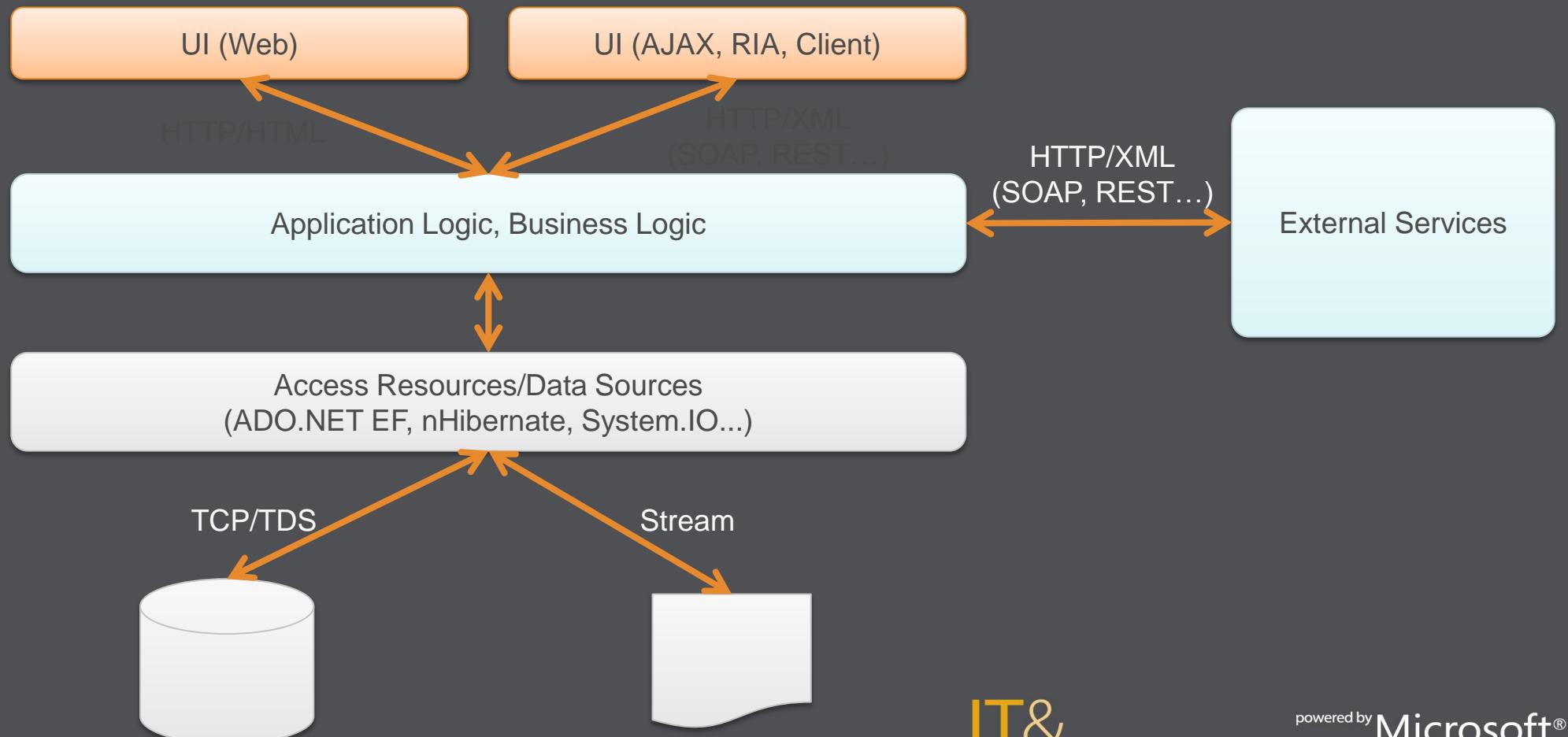
# About The Content

- Combination of publicly available material and my own content
- External Sources
  - Windows Azure Platform Training Kit (WAPTK)
  - TechEd North America 2011 (Channel 9)
  - Windows Azure Bootcamp

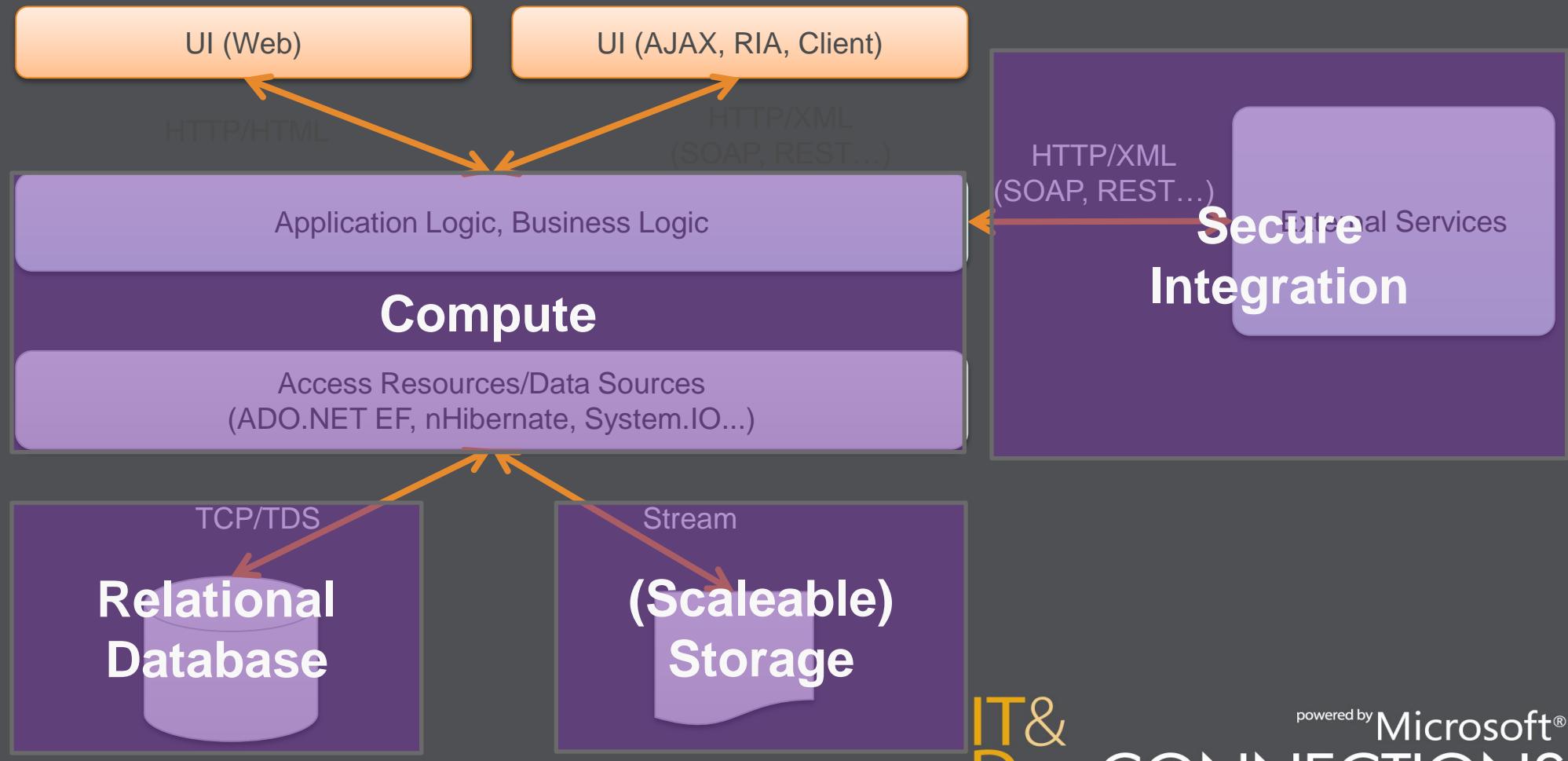
Overview

# STORAGE IN THE WINDOWS AZURE PLATFORM

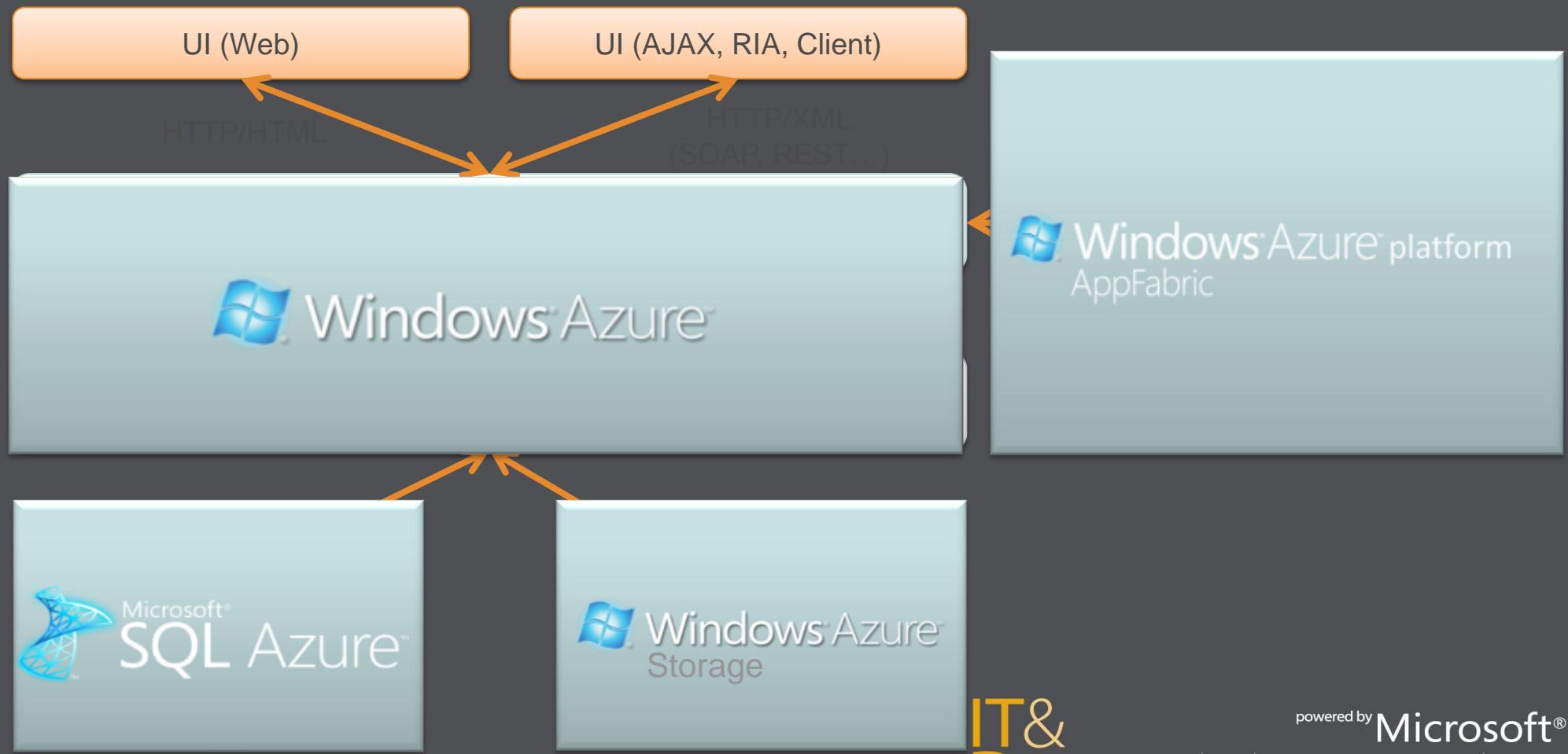
# Structure of a Cloud Platform



# Structure of a Cloud Platform



# Structure of a Cloud Platform



# Windows Azure Storage

- Storage in the Cloud
  - Scalable, durable, and available
  - Anywhere at anytime access
  - Only pay for what the service uses
- Exposed via RESTful Web Services
  - Use from Windows Azure Compute
  - Use from anywhere on the internet
- Various storage abstractions
  - Tables, Blobs, Queues, Drives

# Windows Azure

Your Service



Local Disks  
**Temporary Storage**

Web Portal  
(API)



Fabric  
Controller



Storage Services  
**Persistent Storage**



# When To Use What??

## SQL Azure

- Strong programming model needed
- Need for complex ACID transactions
- Restricted storage amount acceptable (currently max. 50GB/DB)
- TDS is possible

## Windows Azure Storage

- Low price (~1/65<sup>th</sup> compared to SQL Azure)
- Auto-scale out → Fast
- Large storage volumes (many, many TBs)
- REST/HTTP needed
- NTFS needed (Drives)
- Queues needed

# Price Comparison



## Storage

Per GB stored and  
transactions

**\$0.15 GB/month**

\$0.01/10k transactions

## Web Edition

Per database/month

**\$9.99/month**

(1-5 GB DB/month)

## Business Edition

Per database/month

**Starting at \$99.99/month**

(10-50 GB DB/month)

- Think about...
  - ...storage volume needed
  - ...number of transactions
  - ...programming effort
- Background information for Azure Storage Pricing

Introduction

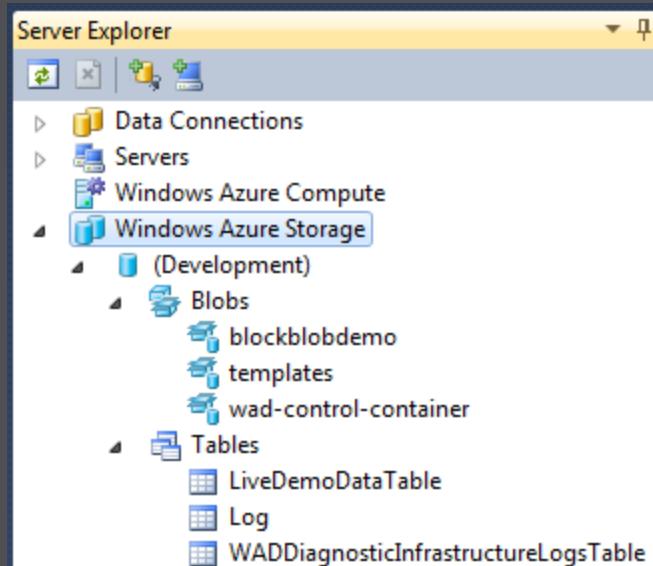
# WINDOWS AZURE STORAGE

# Windows Azure Storage Account

- User specified globally unique account name
  - Can choose geo-location to host storage account
    - US – “North Central” and “South Central”
    - Europe – “North” and “West”
    - Asia – “East” and “Southeast”
  - Can CDN Enable Account
    - More details later
  - Can co-locate storage account with compute account
    - Explicitly or using affinity groups
- Accounts have two independent 512 bit shared secret keys
- 100TB per account

# Tools for Azure Storage

- Server Explorer in Visual Studio 2010
  - Limited functionality
  - Good for developer-specific tasks (e.g. Access IntelliTrace data stored in Azure storage)



# Tools for Azure Storage

- Cerebrata
  - Cloud Storage Studio
  - Azure Diagnostics Manager
- Azure Storage Explorer (Codeplex)
- CloudBerry Labs
  - Explorer for Azure Blob Storage and other cloud storage services
- Gladinet
  - Cloud Storage Tools
- and many more...

# Windows Azure Emulators

- *Windows Azure Compute/Storage Emulator*  
aka DevFabric/DevStorage
  - Port of the [Windows Azure SDK](#) → free
- Provides a local “Mock” storage
  - For debugging purposes
  - To reduce costs
  - To work offline
- Emulator ≠ Windows Azure
  - For differences see [MSDN](#)
    - Important: Predefined account name/key for emulator
  - Testing in emulator alone is not enough!

# Windows Azure Emulators

- Prerequisites
  - [Windows Azure SDK](#) and Azure Tools for VS
  - Visual Studio 2010
  - IIS and SQL Server 2008 R2 (see also [MSDN](#))
- Installation
  - Install SDK and Tools
  - Configure emulator (see also [MSDN](#))
- You cannot access emulators over the network, only local
  - Tip: Use e.g. reverse proxy to access emulators over the network (see e.g. [Emmanuel's Blog](#))

*demo*

# Storage Accounts

Creation and  
Management

# Demo Content

- Storage account management in Azure Portal
- Storage emulator
  - UI
  - Backing storage in local SQL Express
- Account name/key logic
  - Primary/secondary key
- Show Visual Studio Server Explorer
- Show Cerebrata Cloud Storage Studio

# The Storage Client API

- Underlying RESTful API
  - API Reference see [MSDN](#)
  - Can call these from any HTTP client  
e.g. Flash, Silverlight, etc...

- Client API from SDK

*Microsoft.WindowsAzure.StorageClient*

- API Reference see [MSDN](#)
- Provides a strongly typed wrapper around REST services
- <http://azurestoragesamples.codeplex.com/>

# Storage Security

- Windows Azure Storage provides simple security for calls to storage service
  - HTTPS endpoint
  - Digitally sign requests for privileged operations
- Two 512bit symmetric keys per storage account
  - Can be regenerated independently
- More granular security via Shared Access Signatures
  - Details later

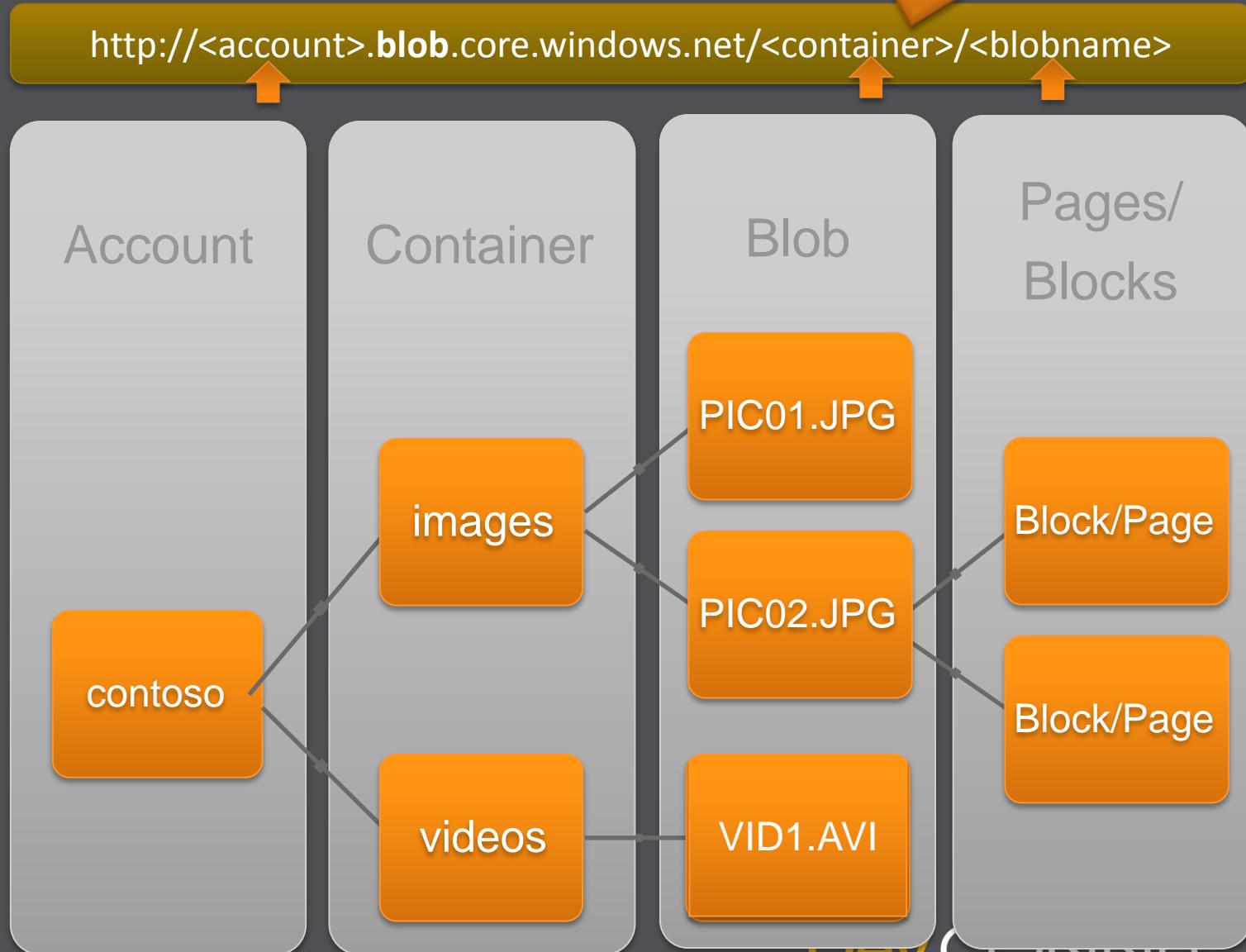
# Windows Azure Storage Types

- **Blobs**
  - Simple named files along with metadata for the file
- **Drives**
  - Durable NTFS volumes for Windows Azure applications to use. Based on Blobs.
- **Tables**
  - Structured storage. A Table is a set of entities; an entity is a set of properties
- **Queues**
  - Reliable storage and delivery of messages for an application

Windows Azure Blob Storage

# BLOB STORAGE

# Blob Storage Concepts



# Blob Containers

- Multiple Containers per Account
  - Special \$root container
- Blob Container
  - A container holds a set of blobs
  - Set access policies at the container level
  - Associate Metadata with Container
    - Key/value pairs
  - List the blobs in a container
    - Including Blob Metadata and MD5
    - NO search/query. i.e. no WHERE MetadataValue = ?
- Blobs Throughput
  - Effectively in Partition of 1
  - Throughput rates similar to disk reads (from within Azure DC)
    - Target of 60MB/s per Blob

# Blob Details

- Main Web Service Operations
  - *PutBlob, GetBlob, DeleteBlob, CopyBlob, SnapshotBlob, LeaseBlob*
  - Complete list see [MSDN](#)
  - Consider garbage collection for delete operations
    - Name reserved until garbage collected
- Associate Metadata with Blob
  - Standard HTTP metadata/headers  
(Cache-Control, Content-Encoding, Content-Type, etc)
  - Metadata is <name, value> pairs, up to 8KB per blob
  - Either as part of PutBlob or independently
- Blob always accessed by name
  - Can include '/' or other delimiter in name  
e.g. /<container>/myblobs/blob.jpg

# Enumerating Blobs

- List Blobs operation takes parameters
  - Prefix
  - Delimiter
  - Include= (snapshots, metadata etc...)

```
http://adventureworks.blob.core.windows.net/  
Products/Bikes/SuperDuperCycle.jpg  
Products/Bikes/FastBike.jpg  
Products/Canoes/Whitewater.jpg  
Products/Canoes/Flatwater.jpg  
Products/Canoes/Hybrid.jpg  
Products/Tents/PalaceTent.jpg  
Products/Tents/ShedTent.jpg
```

# Pagination

- Large lists of Blobs can be paginated
  - Either set maxresults or;
  - Exceed default value for maxresults (5000)

```
http://.../products?comp=list&prefix=Canoes&maxresults=2
```

```
<Blob>Canoes/Whitewater.jpg</Blob>
<Blob>Canoes/Flatwater.jpg</Blob>
<NextMarker>MarkerValue</NextMarker>
```

# demo

## Blob Storage

Accessing blob  
storage through  
managed API

# Demo Content

- Create command line application
- Add Azure SDK references
- Create CloudStorageAccount
  - Show different options
- List blobs in a container
- Upload a blob
- Download a blob

# HowTo: Authenticate

```
// OPTION 1
var account = new CloudStorageAccount(
    new StorageCredentialsAccountAndKey("accountName", "accountkey"),
    true);

// OPTION 2
CloudStorageAccount.SetConfigurationSettingPublisher(
    (configName, setter) =>
    setter(RoleEnvironment.GetConfigurationSettingValue(configName)));
account = CloudStorageAccount.FromConfigurationSetting("settingsName");

// OPTION 3
account = CloudStorageAccount.Parse(
    RoleEnvironment.GetConfigurationSettingValue("settingsName"));
```

# Blob Leases

- Lease Blob operation
  - Returns a lease ID (*x-ms-lease-id* response header)
  - Lease ID has to be passed to subsequent write operations
- Lock a blob for one minute
  - Can be renewed, released and broken
  - Does not lock the container (can be e.g. deleted while there is an active lease)
- Not available in managed API

# HowTo: Acquire Lease

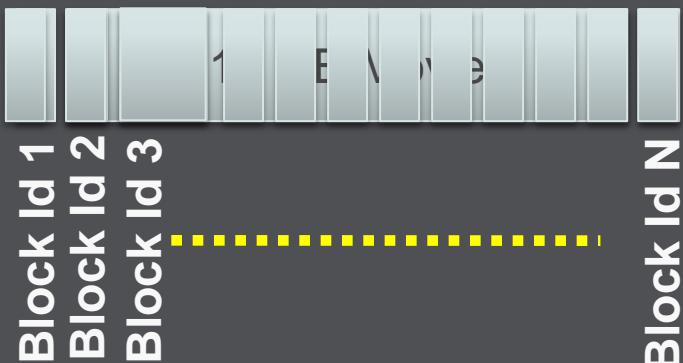
```
public static string AcquireLease(this CloudBlob blob)
{
    var creds = blob.ServiceClient.Credentials;
    var transformedUri = new Uri(creds.TransformUri(blob.Uri.ToString()));
    var req = BlobRequest.Lease(transformedUri,
        90, // timeout (in seconds)
        LeaseAction.Acquire, // as opposed to "break" "release" or "renew"
        null); // name of the existing lease, if any
    blob.ServiceClient.Credentials.SignRequest(req);
    using (var response = req.GetResponse())
    {
        return response.Headers["x-ms-lease-id"];
    }
}
```

# Two Types of Blobs Under the Hood

- Block Blob
  - Targeted at streaming workloads
  - Each blob consists of a sequence of blocks
    - Each block is identified by a Block ID
    - Uncommitted blocks are garbage collected after a week
  - Size limit 200GB per blob
  - Optimistic Concurrency via ETags
- Page Blob
  - Targeted at random read/write workloads
  - Each blob consists of an array of pages
    - Each page is identified by its offset from the start of the blob
  - Size limit 1TB per blob
  - Optimistic or Pessimistic (locking) concurrency via Leases

# Uploading a Block Blob

- Uploading a large blob



- Benefit
  - Efficient continuation and retry
  - Parallel and out of order upload of blocks

```
blobName = "TheBlob.wmv";
PutBlock(blobName, blockId1,
block1Bits);
PutBlock(blobName, blockId2,
block2Bits);
.....
PutBlock(blobName, blockIdN,
blockNBits),
PutBlockList(blobName,
blockId1,...,blockIdN);
```

TheBlob.wmv

Windows  
Azure Storage

# HowTo: Upload Block Blobs

```
// Create a block blob consisting of three "files" (=blocks)
var blobRef = container.GetBlockBlobReference("MyBlob.dat");
string xmlBlockID, file1BlockID, file2BlockID;
blobRef.PutBlock(
    xmlBlockID = Convert.ToBase64String(new byte[] { 0 }),
    new MemoryStream(Encoding.UTF8.GetBytes("<root>...</root>")),
    null);
blobRef.PutBlock(
    file1BlockID = Convert.ToBase64String(new byte[] { 1 }),
    new MemoryStream(Encoding.UTF8.GetBytes("This is a test file!")),
    null);
blobRef.PutBlock(
    file2BlockID = Convert.ToBase64String(new byte[] { 2 }),
    new MemoryStream(Encoding.UTF8.GetBytes("This is a second test file!")),
    null);

// Commit creation of all "files" (=blocks)
blobRef.PutBlockList(new[] { xmlBlockID, file1BlockID, file2BlockID });
```

# HowTo: Change Single Block

```
// Change a single "file" (=block) of the blob.  
// Note that this shows how to upload a specific "file" (=block)  
// without uploading the entire blob.  
blobRef = container.GetBlockBlobReference("MyBlob.dat");  
dictation.PutBlock(  
    file1BlockID,  
    new MemoryStream(Encoding.UTF8.GetBytes("This is a CHANGED test file!")),  
    null);  
  
// Display current content of the "file" (=block).  
// Note that the content has NOT changed yet because the upload  
// has not been committed.  
var blobRef2 = container.GetBlockBlobReference("MyBlob.dat");  
Console.WriteLine(DownloadTextFromBlob(blobRef2, 1));  
  
// Now commit the change  
UpdateBlobBlocks(blobRef);
```

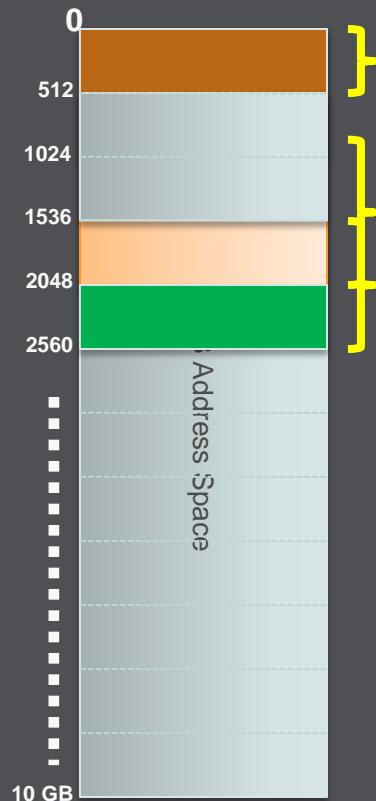
# HowTo: Change Single Block

```
public static string DownloadTextFromBlob(CloudBlockBlob blob, int blockIndex)
{
    var blocks = blob.DownloadBlockList();
    var transformedUri = new Uri(blob.ServiceClient.Credentials.TransformUri(
        blob.Uri.ToString()));
    var webRequest = BlobRequest.Get(transformedUri, 90, null,
        blocks.Take(blockIndex).Aggregate<ListBlockItem, long>(0,
            (agg, item) => agg + item.Size),
        blocks.Skip(blockIndex).First().Size, null);
    blob.ServiceClient.Credentials.SignRequest(webRequest);
    using (var response = webRequest.GetResponse())
    {
        using (var responseStream = response.GetResponseStream())
        {
            var buffer = new byte[blocks.Skip(blockIndex).First().Size];
            responseStream.Read(buffer, 0, buffer.Length);
            return Encoding.UTF8.GetString(buffer);
        }
    }
}
```

# HowTo: Change Single Block

```
public static void UpdateBlobBlocks(CloudBlockBlob blob)
{
    var transformedUri = new Uri(blob.ServiceClient.Credentials.TransformUri(
        blob.Uri.ToString()));
    var webRequest = BlobRequest.PutBlockList(transformedUri, 90, new BlobProperties(),
        null);
    var blocks = blob.DownloadBlockList(new BlobRequestOptions()
    { BlobListingDetails = BlobListingDetails.UncommittedBlobs });
    var content = new StringBuilder();
    content.Append(@"<?xml version=""1.0"" encoding=""utf-8""?><BlockList>");
    foreach (var block in blocks) {
        content.Append("<Latest>");
        content.Append(block.Name);
        content.Append("</Latest>");
    }
    content.Append("</BlockList>");
    using (var stream = webRequest.GetRequestStream()) {
        byte[] contentBytes;
        stream.Write(contentBytes = Encoding.UTF8.GetBytes(content.ToString())),
            0, contentBytes.Length);
    }
    blob.ServiceClient.Credentials.SignRequest(webRequest);
    using (var response = webRequest.GetResponse()) { /* Process result */ }
}
```

# Page Blob – Random Read/Write



- Create MyBlob
  - Specify Blob Size = 10 Gbytes
  - Sparse storage - Only charged for pages with data stored in them
- Fixed Page Size = 512 bytes
- Random Access Operations
  - PutPage[512, 2048)
  - PutPage[0, 1024)
  - ClearPage[512, 1536)
  - PutPage[2048,2560)
- GetPageRange[0, 4096)  
returns valid data ranges:
  - [0,512) , [1536,2560)
- GetBlob[1000, 2048) returns
  - All 0 for first 536 bytes
  - Next 512 bytes are data stored in [1536,2048)

# Blob Security

- Full public read access
  - Anonymous reading of blobs
  - Anonymous enumeration of blobs
- Public read access for blobs only
  - Anonymous reading of blobs
- No public read access
- More granular control with Shared Access Signatures

# Shared Access Signatures

- Fine grain access rights to blobs and containers
- Sign URL with storage key – permit elevated rights
- Revocation
  - Use short time periods and re-issue
  - Use container level policy that can be deleted
- Two broad approaches
  - Ad-hoc
    - Note: Max. duration one hour
  - Policy based

# Ad Hoc Signatures

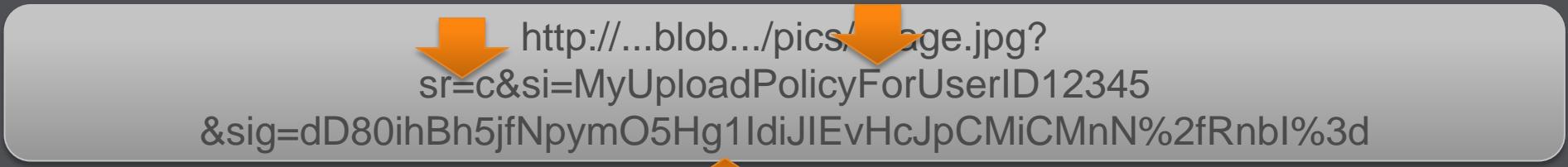
- Create Short Dated Shared Access Signature
  - Signedresource Blob or Container
  - AccessPolicy Start, Expiry and Permissions
  - Signature HMAC-SHA256 of above fields

http://...blob.../pics/image  
sr=c&st=2009-02-09T08:20Z&se=2009-02-10T08:30Z&sp=w  
&sig= dD80ihBh5jfNpymO5Hg1ldiJIEvHcJpCMiCMnN%2fRnbI%3d

- Use case
  - Single use URLs
  - E.g. Provide URL to Silverlight client to upload to container

# Policy Based Signatures

- Create Container Level Policy
  - Specify StartTime, ExpiryTime, Permissions
- Create Shared Access Signature URL
  - Signedresource Blob or Container
  - Signedidentifier Optional pointer to container policy
  - Signature HMAC-SHA256 of above fields



http://...blob.../pics/page.jpg?  
sr=c&si=MyUploadPolicyForUserID12345  
&sig=dD80ihBh5jfNpymO5Hg1ldiJIEvHcJpCMiCMnN%2fRnbI%3d

An orange arrow points down to the URL path 'page.jpg?'. Another orange arrow points down to the query parameter 'sr=c&si=MyUploadPolicyForUserID12345'. A third orange arrow points up from the bottom of the URL to the '&sig=' part.

- Use case
  - Providing revocable permissions to certain users/groups
  - To revoke: Delete or update container policy

# demo

## Shared Access Signatures

Securing blobs with  
signatures

# Demo Content

- Ad hoc SAS in Cerebrata
- Container policy in Cerebrata
  - Default policy
  - Show use of policy

# HowTo: Create Container Policy

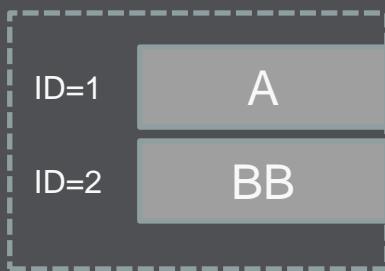
```
public bool SetContainerAccessPolicy(string containerName,
    SortedList<string, SharedAccessPolicy> policies)
{
    try {
        CloudBlobContainer container = BlobClient.GetContainerReference(containerName);
        BlobContainerPermissions permissions = container.GetPermissions();
        permissions.SharedAccessPolicies.Clear();
        if (policies != null) {
            foreach (KeyValuePair<string, SharedAccessPolicy> policy in policies)
            {
                permissions.SharedAccessPolicies.Add(policy.Key, policy.Value);
            }
        }
        container.SetPermissions(permissions);
        return true;
    }
    catch (StorageClientException ex) {
        if ((int)ex.StatusCode == 404) {
            return false;
        }
        throw;
    }
}
```

# What Are Snapshots?

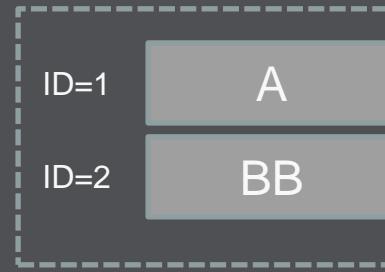
- Create a point in time read-only copy of a blob
- Every snapshot creates a new read only point in time copy
- Restore snapshots using copy blob
- Cleanup your snapshots
- Charged only for unique blocks or pages i.e. reuse blocks or pages
  - For reuse, use WritePages or PutBlock & PutBlockList

# What does unique mean?

1. Add 2 blocks {1, 2}
2. Commit the blob
3. Take a snapshot
4. Charged for
  - Base blob's blocks {1, 2}



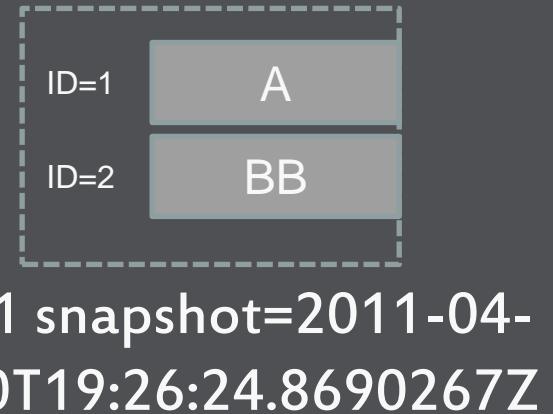
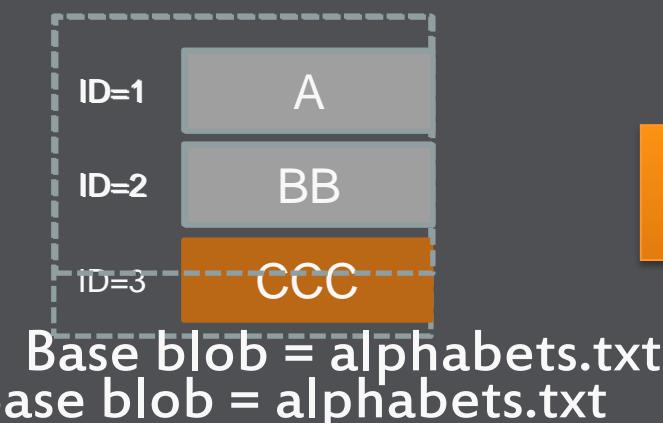
Base blob = alphabets.txt



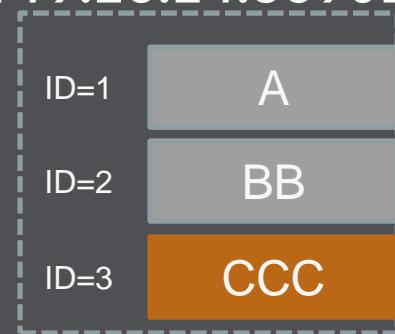
#1 snapshot=2011-04-10T19:26:24.8690267Z

# What does unique mean?

5. Add a new block Id=3
6. Commit the base blob
7. Charged for
  - Base blob's blocks {1, 2, 3}
8. Snapshot to create the second snapshot
9. Charged for
  - Base blob's blocks {1, 2, 3}

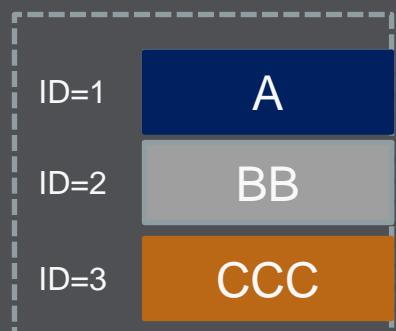


#2 snapshot=2011-05-10T19:26:24.8690267Z



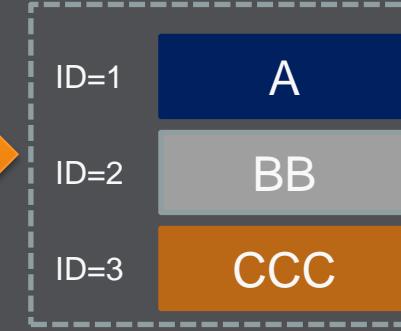
# What does unique mean?

9. Update block id 1 to now contain the same data "A"
10. Commit the blob
11. Charged for:
  - Base blob's blocks {1, 2, 3}
  - Snapshot #1's blocks {1}
12. Take a snapshot
13. Charged for:
  - Base blob's blocks {1, 2, 3}
  - Snapshot #1's blocks {1}

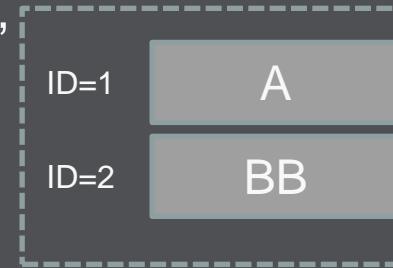


Base blob = alphabets.txt

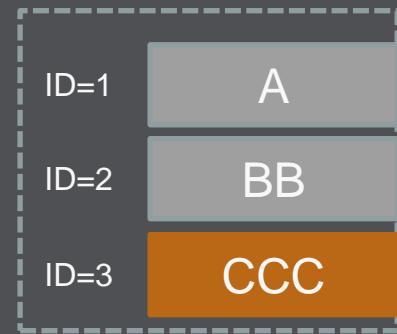
Snapshot #3



#3 snapshot=2011-05-10T19:28:24.8690267Z



#1 snapshot=2011-04-10T19:26:24.8690267Z



#2 snapshot=2011-05-10T19:26:24.8690267Z

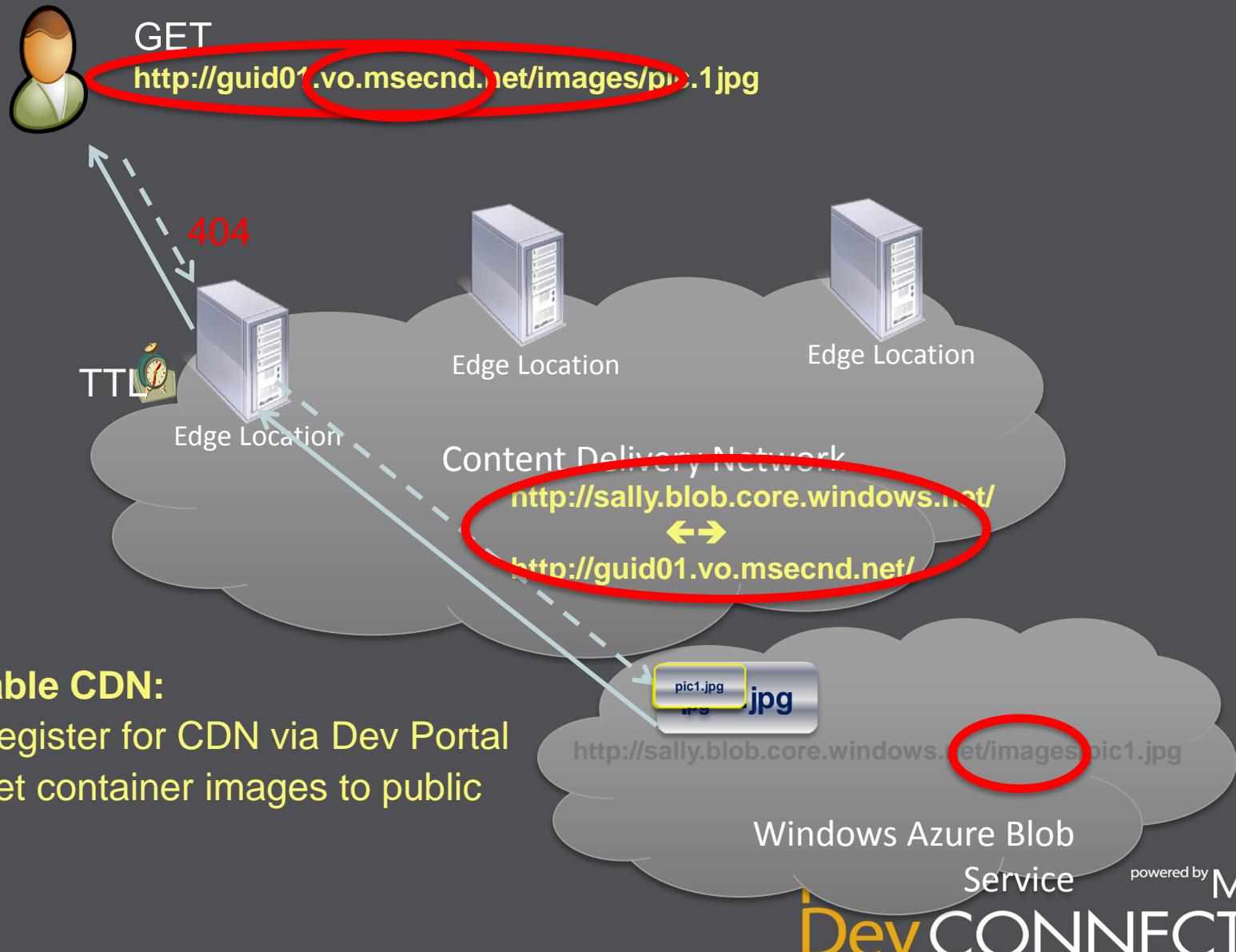
# HowTo: Create Snapshot

```
CloudBlobContainer container =  
    BlobClient.GetContainerReference(containerName);  
CloudBlob blob = container.GetBlobReference(blobName);  
blob.CreateSnapshot();
```

# Content Delivery Network (CDN)

- Scenario
  - Frequently accessed blobs
  - Accessed from around the world
- Windows Azure Content Delivery Network (CDN) provides high-bandwidth global blob content delivery
  - 20 locations globally (US, Europe, Asia, Australia and South America), and growing
  - Same experience for users no matter how far they are from the geo-location where the storage account is hosted
- Blob service URL vs CDN URL:
  - Windows Azure Blob URL: <http://images.blob.core.windows.net/>
  - Windows Azure CDN URL: <http://<id>.vo.msecnd.net/>
  - Custom Domain Name for CDN: <http://cdn.contoso.com/>
- Cost
  - US located CDN nodes 15c/GB + 1c/10,000 txn
  - Rest of World 20c/GB + 1c/10,000 txn
  - Traffic from Storage node to edge node at standard rates

# Azure Content Delivery Network



# demo

## CDN

### Content Delivery Network

# Demo Content

- Static website served from Blob Storage
- CDN configuration
- Static website served from CDN

# Tips & Tricks for Blobs

- Performance best practices see [Azurescope](#)
- No need to rewrite existing *System.IO*-based applications
  - Use Drives instead of raw blob access (see next chapter)
- Upload/download large blobs in chunks in parallel
- Use snapshot feature for backup purposes
- Combine blobs into containers intelligently
  - E.g. bulk deletes for logs
- Intelligently combine different storage mechanisms
  - SQL Azure for structured data, store blobs in Azure Blob storage
  - Note: No distributed transactions from SQL Azure to Windows Azure Storage
- Keep garbage collection in mind

Windows Azure Drives

# DRIVES

# Windows Azure Drives

- Durable NTFS volume for Windows Azure Instances
  - Use existing NTFS APIs to access a network attached durable drive
  - Use *System.IO* from .NET
- Benefits
  - Move existing apps using NTFS more easily to the cloud
  - Durability and survival of data on instance recycle
- A Windows Azure Drive is a NTFS VHD Page Blob
  - Mounts Page Blob over the network as an NTFS drive
  - Local cache on instance for read operations
  - All flushed and unbuffered writes to drive are made durable to the Page Blob

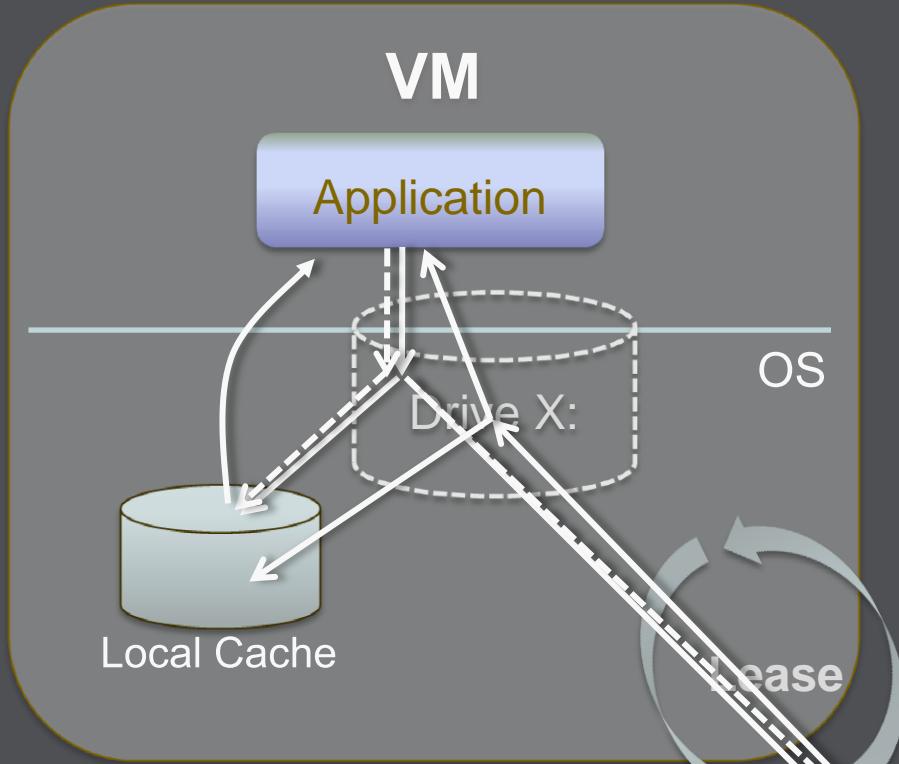
# Windows Azure Drive Capabilities

- A Windows Azure Drive is a Page Blob formatted as a NTFS single volume Virtual Hard Drive (VHD)
  - Drives can be up to 1TB
- A Page Blob can be mounted:
  - On **one instance** at a time for read/write
  - Using read-only snapshots to multiple instances at once
- An instance can dynamically mount up to 16 drives
- Remote Access via standard BlobUI
  - Can't remotely mount drive.
  - Can upload the VHD to a Page Blob using the blob interface, and then mount it as a Drive
  - Can download the VHD to a local file and mount locally

# Drive Details

- Operations performed via Drive API not REST Calls
- Operations on Drives
  - **CreateDrive**
    - Creates a new NTFS formatted VHD in Blob storage
  - **MountDrive/UnmountDrive**
    - Mounts a drive into Instance at new drive letter
    - Unmounts a drive freeing drive letter
  - **Get Mounted Drives**
    - List mounted drives; underlying blob and drive letter
  - **Snapshot Drive**
    - Create snapshot copy of the drive

# Windows Azure Drives



- Drive is a formatted page blob stored in blob service
- Mount obtains a blob lease
- Mount specifies amount of local storage for cache
- NTFS flushed/unbuffered writes commit to blob store before returning to app
- NTFS reads can be served from local cache or from blob store (cache miss)

DemoBlob

Windows Azure  
Blob Service

Microsoft®  
DEV CONNECTIONS

# Cloud Drive Client Library Sample

```
CloudStorageAccount account =
    CloudStorageAccount.FromConfigurationSetting("CloudStorageAccount");

CloudDrive.InitializeCache(localCacheDir, cacheSizeInMB);
CloudDrive drive = account.CreateCloudDrive(pageBlobUri);
drive.Create(1000 /* sizeInMB */);

string pathOnLocalFS = drive.Mount(cacheSizeInMB,
    DriveMountOptions.None);

//Use NTFS APIs to Read/Write files to drive

//Snapshot drive while mounted to create backups
Uri snapshotUri = drive.Snapshot();

drive.Unmount();
```

# Failover with Drives

- Must issue NTFS Flush command to persist data
  - Use *System.IO.Stream.Flush()*
- Read/Write Drives protected with leases
  - 1 Minute lease expiry
  - Maintained by Windows Azure OS Driver
  - *Unmount* on *RoleEntryPoint.OnStop*
- On failure
  - Lease will timeout after 1 minute
  - Re-mount drive on new instance

# demo

## Drives

### Windows Azure Drives

# Demo Content

- WAPTK Lab „Exercise 4: Working with Drives“
- Show Windows Azure Drives in Storage Emulator
- Create VHD with Windows 7
- Upload VHD

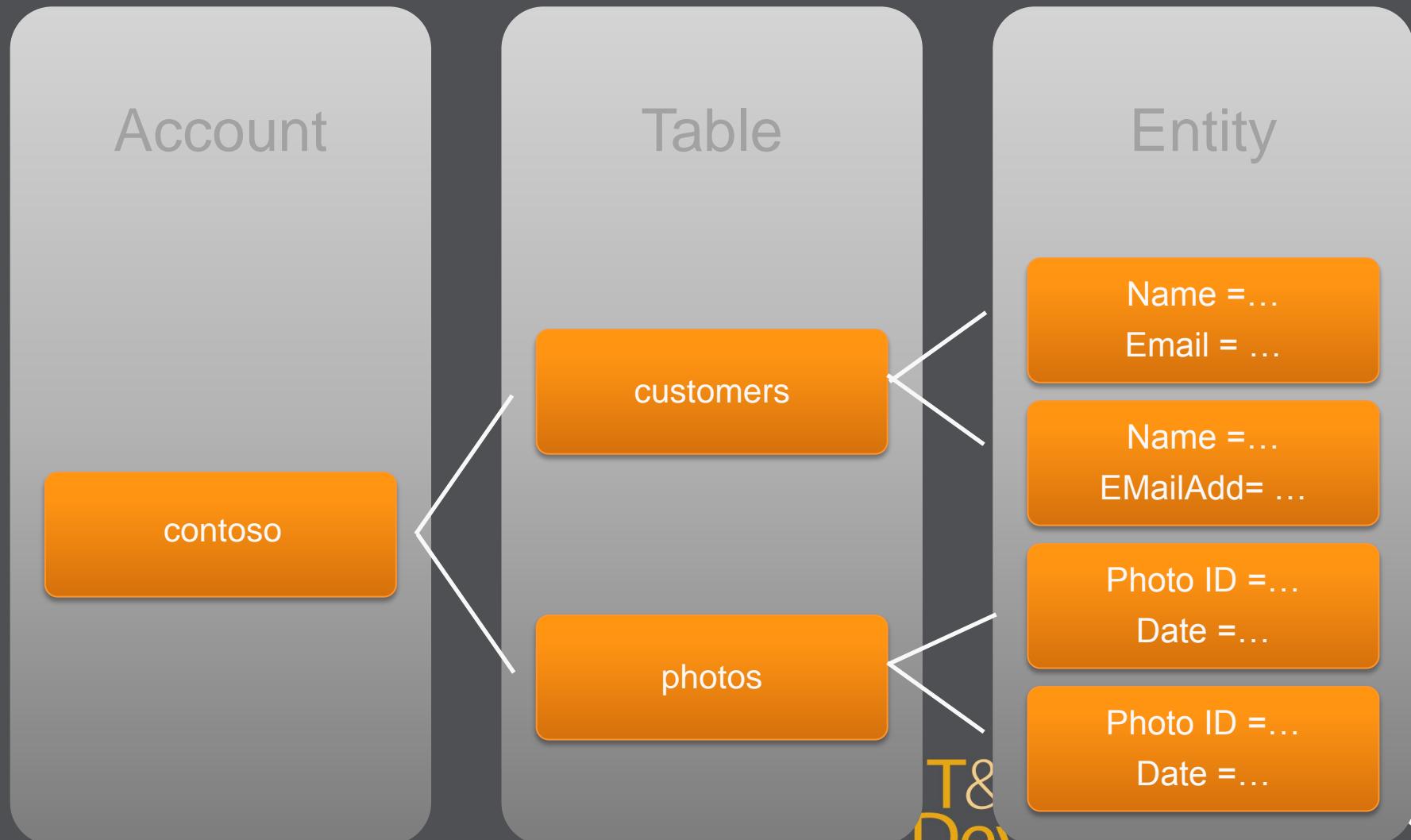
Windows Azure Table Store

# TABLES

# Tables

- NoSQL, not an RDBMS!
  - Limited transactions
  - No foreign keys
  - Only a single index/table
  - No stored procs/funcs
  - Limited query capabilities
  - Etc.
- OData protocol is the native protocol
  - From .NET: WCF Data Services → LINQ

# Table Storage Concepts



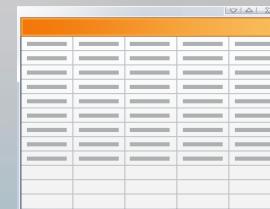
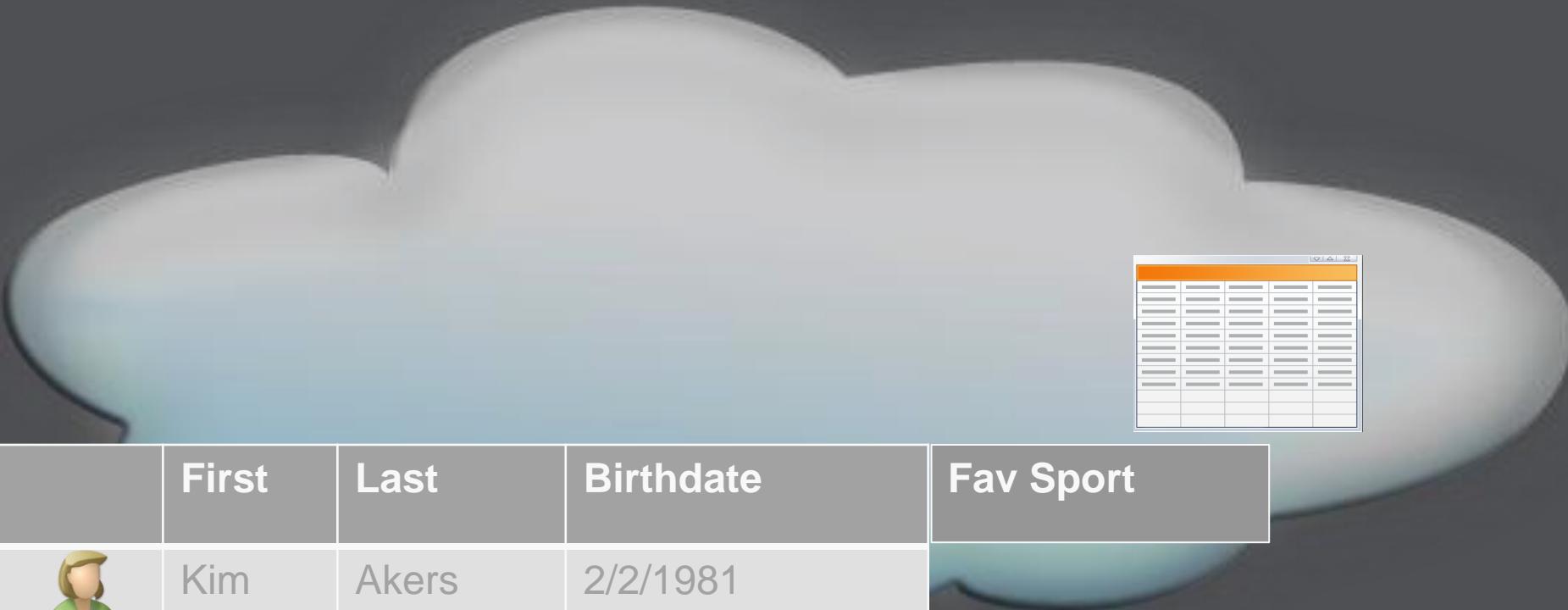
# Table Operations

- **Table**
  - **Create, Query, Delete**
  - Tables can have metadata
- **Entities**
  - **Insert**
  - **Update**
    - Merge – Partial update
    - Replace – Update entire entity
  - **Delete**
  - **Query**
  - Entity Group Transactions
    - Multiple CUD Operations in a single atomic transaction

# Entity Properties

- Entity can have up to 255 properties
  - Up to 1MB per entity
- Mandatory Properties for every entity
  - PartitionKey & RowKey (string; only indexed properties)
  - Uniquely identifies an entity
  - Defines the sort order
  - Timestamp
  - Optimistic Concurrency. Exposed as an HTTP ETag
- No fixed schema for other properties
  - Each property is stored as a <name, typed value> pair
  - No schema stored for a table
  - Properties can be the standard .NET types
  - *string, binary, bool, DateTime, GUID, int, int64, and double*

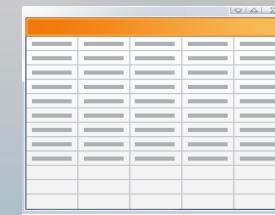
# No Fixed Schema



	First	Last	Birthdate	Fav Sport
	Kim	Akers	2/2/1981	
	Nancy	Anderson	3/15/1965	Canoeing
	Mark	Hassall	May 1, 1976	

# Querying

?\$filter=Last eq 'Akers'

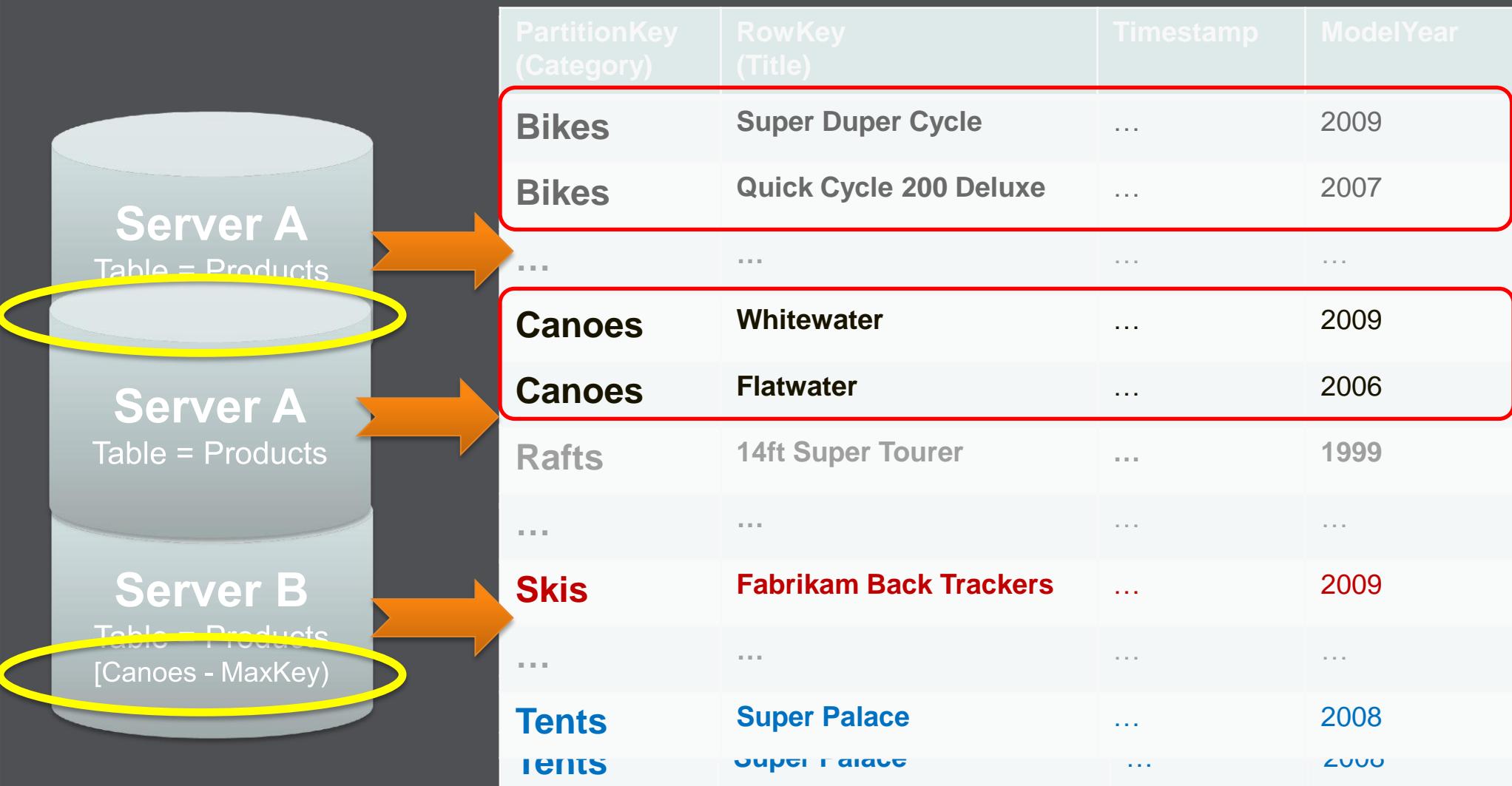


	First	Last	Birthdate
	Kim	Akers	2/2/1981
	Nancy	Anderson	3/15/1965
	Mark	Hassall	May 1, 1976

# Purpose of the PartitionKey

- Entity Locality
  - Entities in the same partition will be stored together
    - Efficient querying and cache locality
    - Endeavour to include partition key in all queries
- Entity Group Transactions
  - Atomic multiple Insert/Update/Delete in same partition in a single transaction
- Table Scalability
  - Target throughput – 500 tps/partition, several thousand tps/account
  - Windows Azure monitors 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

# Partitions and Partition Ranges



# demo

## Tables

### Windows Azure Table Store

# Demo Content

- Table management with Cerebrata
- If necessary
  - Intro to OData with WCF data services and Fiddler
- WPF demo application
  - Similar to WAPTK „Exercise 1: Working with Tables“
  - Create table
  - Add some rows (demo batching)
  - Simple queries with LINQ

# Tables – Best Practices & Tips

- Use clustered index in queries for performance
  - Filter on partition key/row key
- Limit large scans
- Expect continuation tokens for queries that scan
  - Split “OR” on keys as individual queries
- Point query throws an exception if resource does not exist.
  - *DataServiceContext.IgnoreResourceNotFoundException = true*
- For “read only” scenarios turn off tracking
  - *DataServiceContext.MergeOption = MergeOption.NoTracking*
- Follow performance tips by Microsoft

# Tables – Best Practices & Tips

- Entity Group Transaction
  - Get transaction semantics
- Note that property names are stored for each entity
  - Influences amount of storage you have to pay for
- Do not reuse *DataServiceContext* across multiple logical operations and discard on failures
- *AddObject/AttachTo* can throw exception if entity is already being tracked
- Replicate data instead of joins
  - Remember, storage is cheap

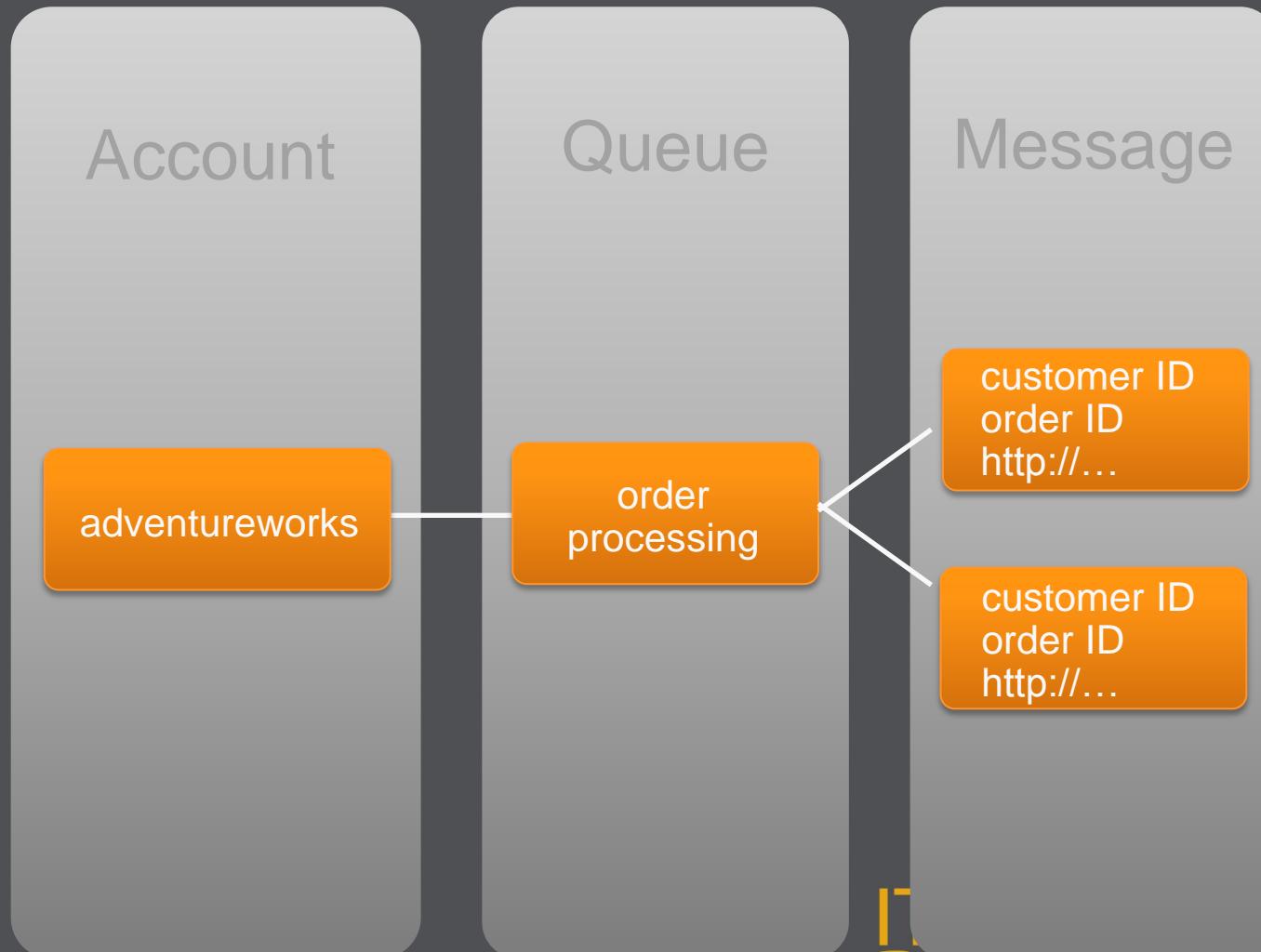
# Tables – Best Practices & Tips

- Keep garbage collection in mind
  - Delete operations are done asynchronously
- Reduce costs with batching

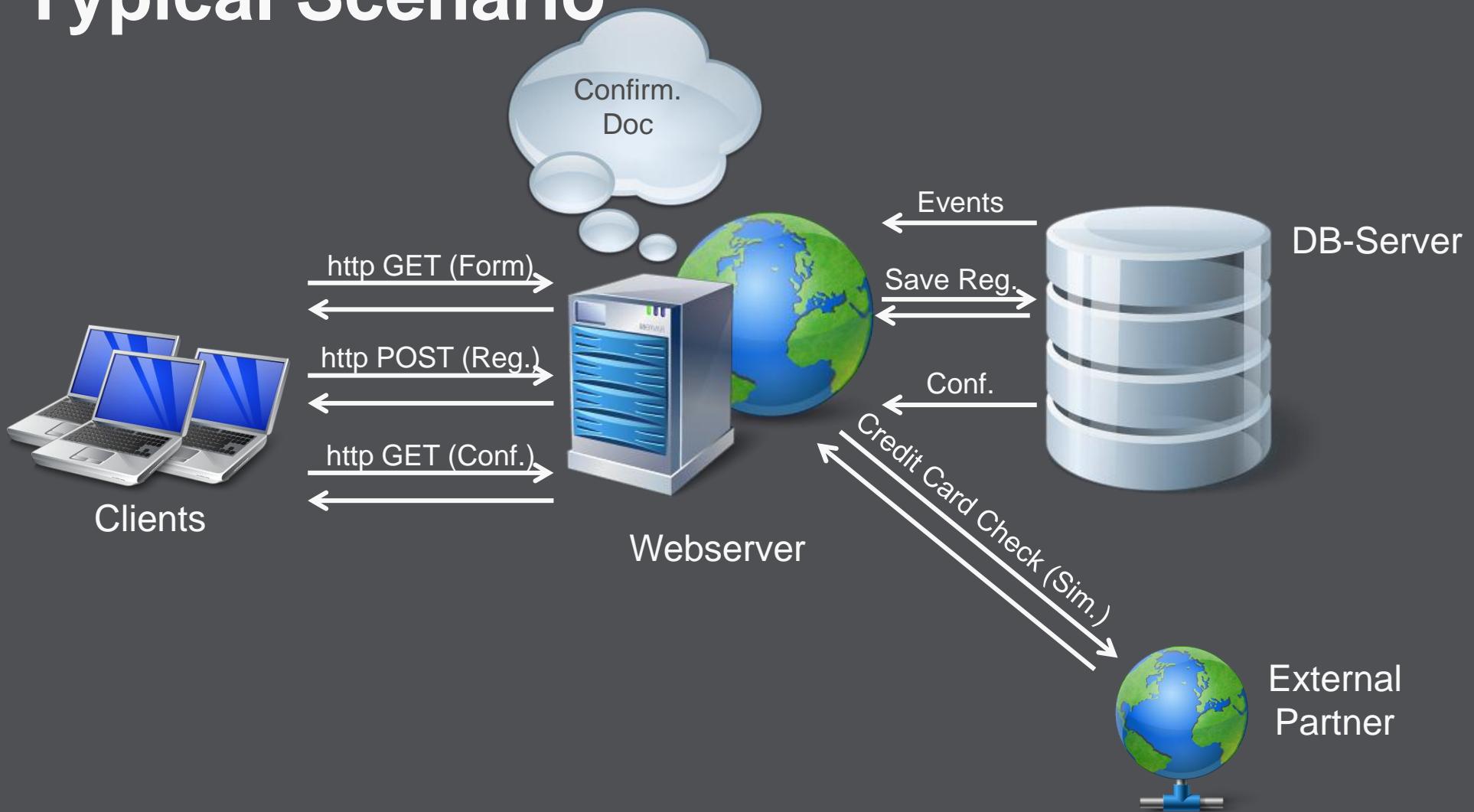
Windows Azure Queues

# QUEUES

# Queue Storage Concepts



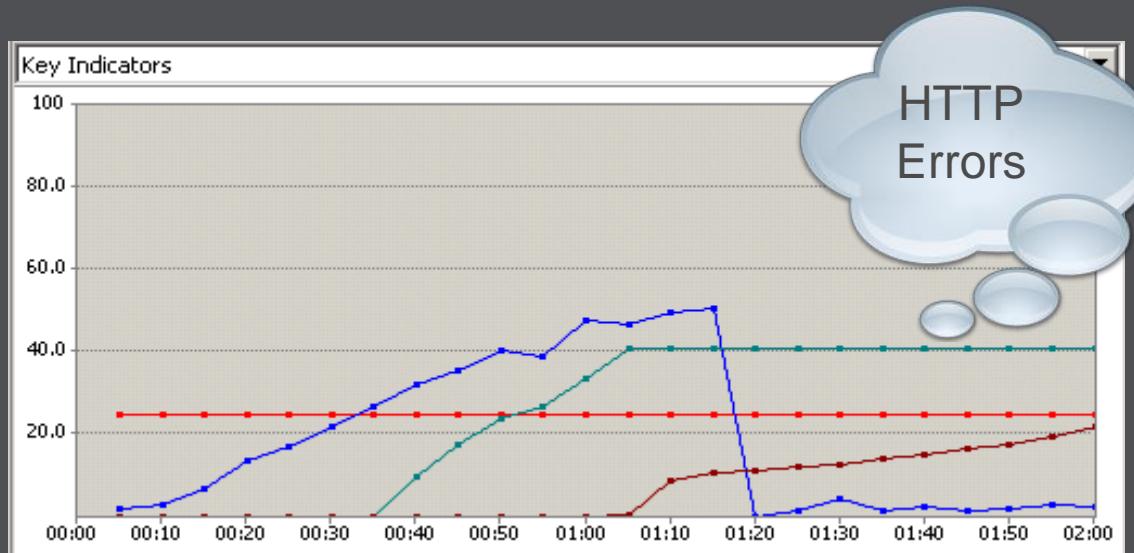
# Typical Scenario



# Typical Scenario

- Webserver is blocked by long running requests
- Web server's CPU is not the bottleneck
  - Dependency on external service (Credit card check)
  - Dependency on database (Processing time, locking)
- Doesn't scale at all!

# Results – Processing Time

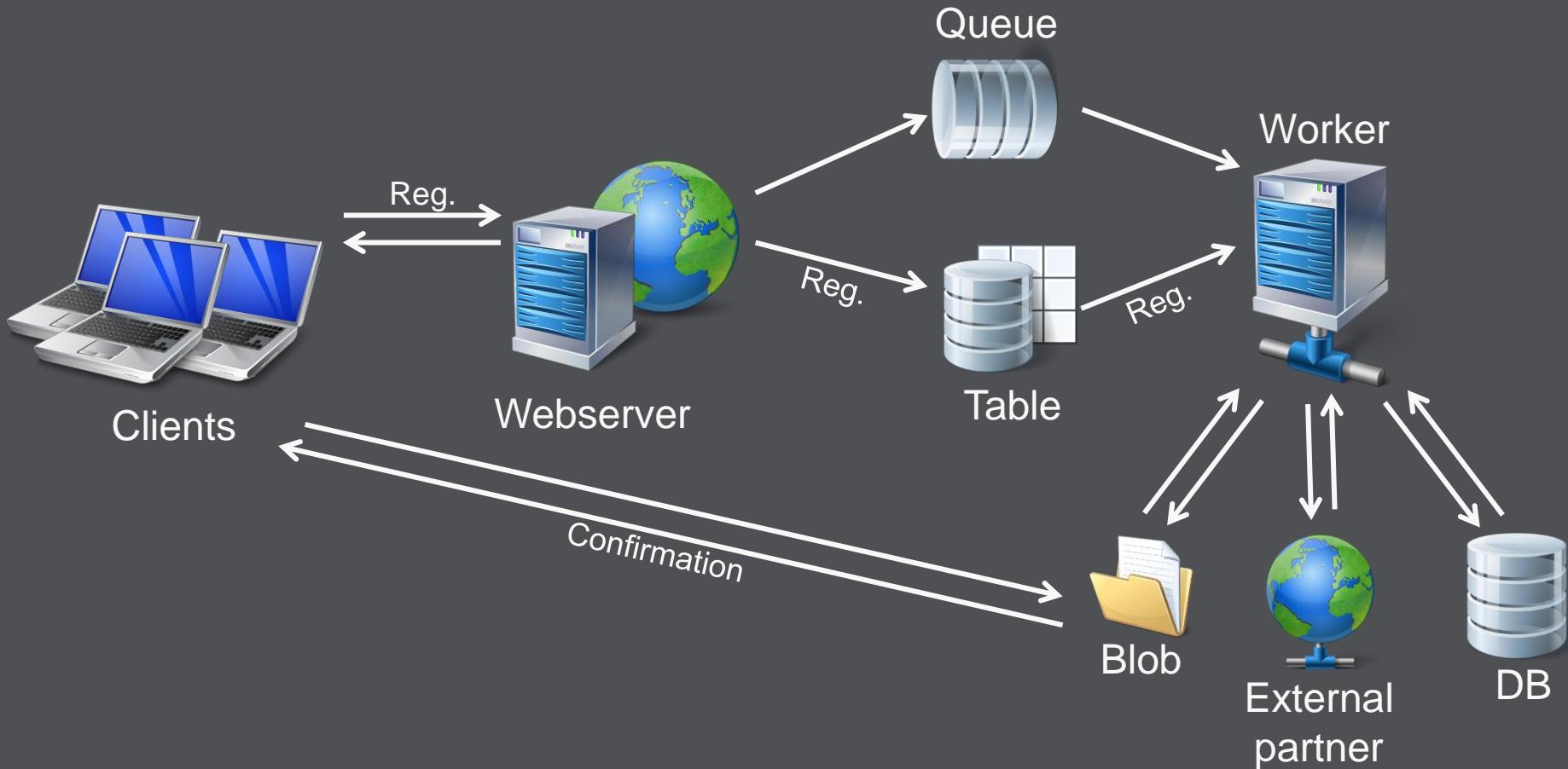


Counter	Instance	Category	Computer	Color	Range	Min	Max	Avg
<b>Key Indicators</b>								
<input checked="" type="checkbox"/> User Load	_Total	LoadTest:Scenario	IP-0AE33654	<span style="color:red">█</span>	1,000	250	250	250
<input checked="" type="checkbox"/> Avg. Page Time	_Total	LoadTest:Page	IP-0AE33654	<span style="color:blue">█</span>	100	0.41	51.1	14.4
<input checked="" type="checkbox"/> Http Errors	_Total	LoadTest:Errors	IP-0AE33654	<span style="color:green">█</span>	100	0	41	25
<input checked="" type="checkbox"/> Exceptions	_Total	LoadTest:Errors	IP-0AE33654	<span style="color:red">█</span>	1,000	0	222	68

# Results – CPU Utilization



# Async Scenario With Queues



```

ClientScript.GetPostBackEventReference(this, string.Empty);

if (!Page.IsPostBack)
{
    Guid registrationId = Guid.NewGuid();

    Save time when request has arrived

    Build registration object from http parameters

#region Add registration in processing queue
// Add registration request to queue
cloudStorage.RegistrationQueue.AddMessage(new CloudQueueMessage(registration.RegistrationId.ToString()));

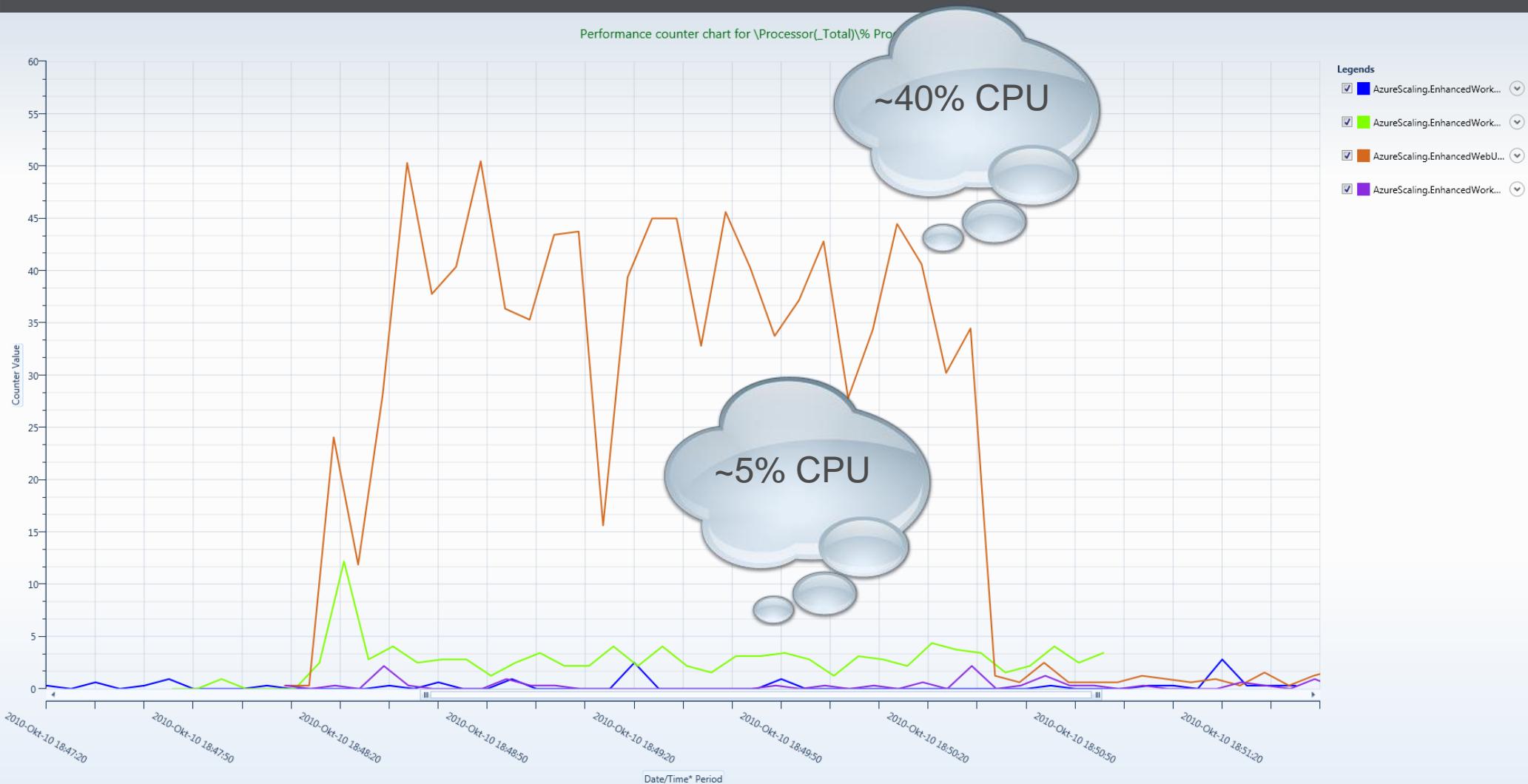
// Add registration payload to table
cloudStorage.RegistrationPayloadContext.AddObject(CloudStorageConnection.RegistrationTableName, registration);
cloudStorage.RegistrationPayloadContext.SaveChanges();
#endregion

this.RegistrationId.Value = registrationId.ToString();
this.WelcomeMessage.Text = "Registration is currently processed, will take a while...";
RegisterRefreshScript();
}
else
{
    Display registration status
}

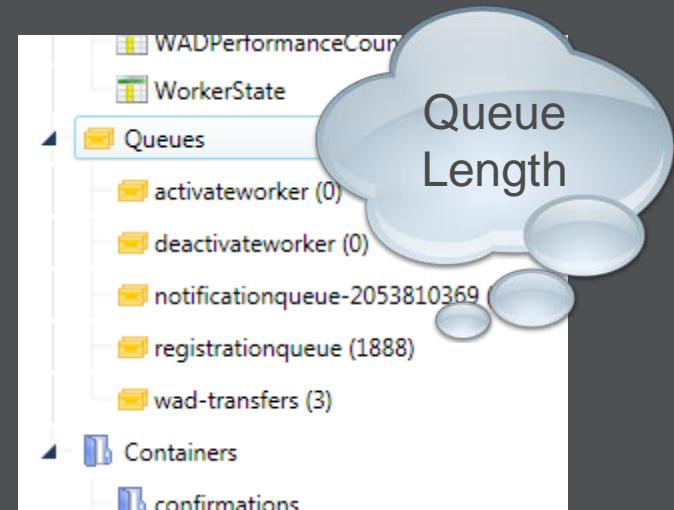
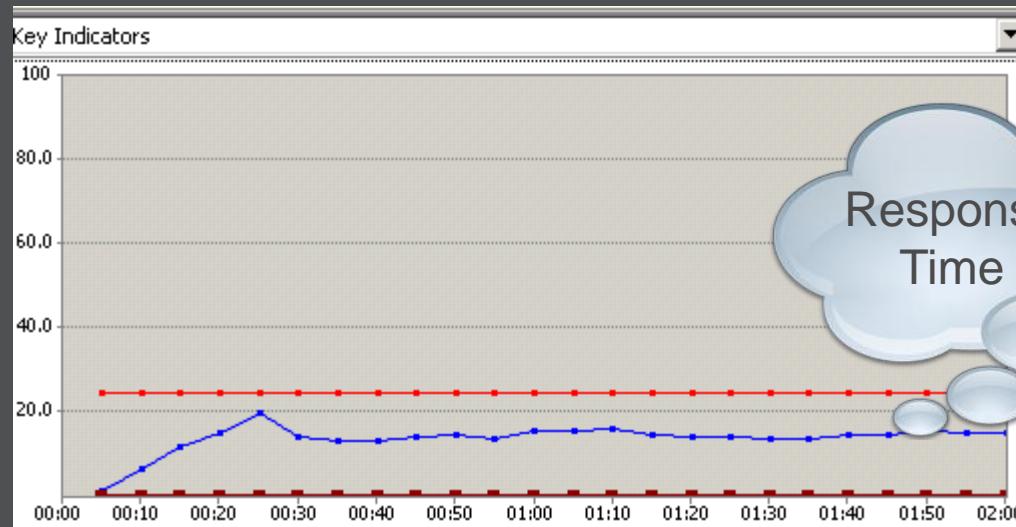
```



# Results – CPU Utilization



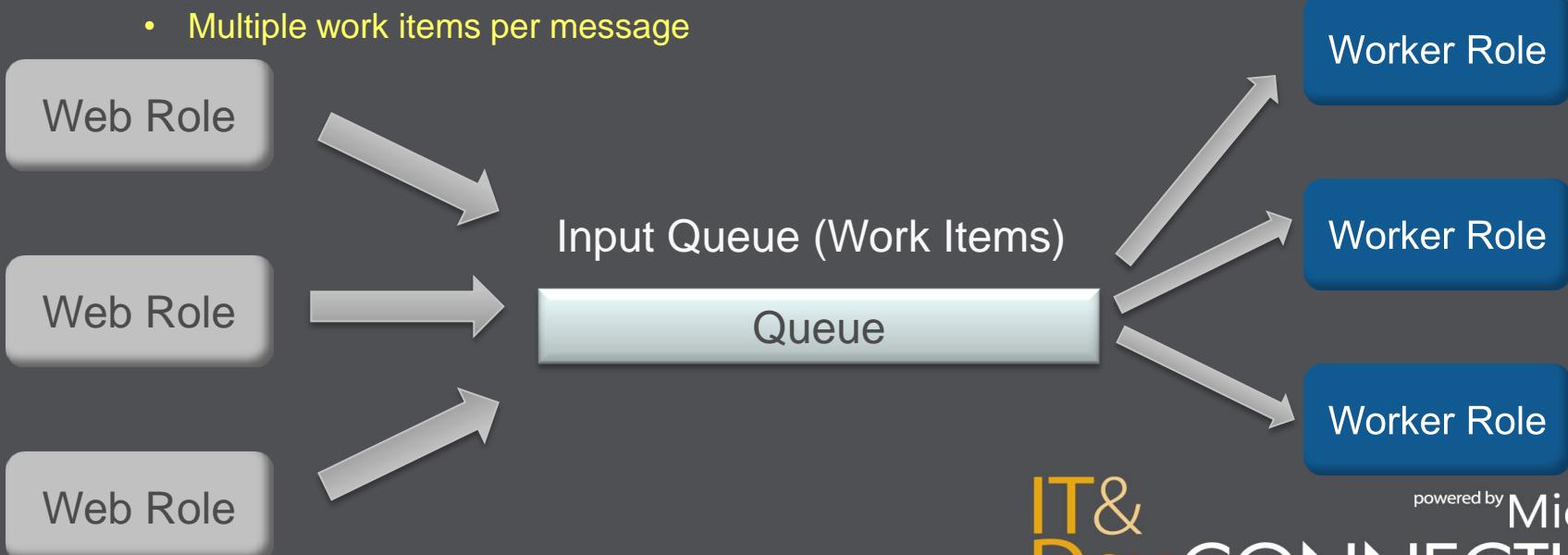
# Results – Processing Time



Counter	Instance	Category	Computer	Color	Range	Min	Max	Avg
<b>Key Indicators</b>								
<input checked="" type="checkbox"/> User Load	_Total	LoadTest:Scenario	IP-OAE33654		1,000	250	250	250
<input checked="" type="checkbox"/> Avg. Page Time	_Total	LoadTest:Page	IP-OAE33654		100	1.86	20.3	14.9
<input checked="" type="checkbox"/> Exceptions	_Total	LoadTest:Errors	IP-OAE33654		0	0	0	0
<input checked="" type="checkbox"/> Http Errors	_Total	LoadTest:Errors	IP-OAE33654		0	0	0	0

# Loosely Coupled Workflow with Queues

- Enables workflow between roles
  - Load work in a queue
    - Producer can forget about message once it is in queue
  - Many workers consume the queue
  - For extreme throughput (>500 tps)
    - Use multiple queues
    - Read messages in batches
    - Multiple work items per message



# Queue Details

- Simple asynchronous dispatch queue
  - No limit to queue length subject to storage limit
  - 8kb per message
  - **ListQueues** - List queues in account
- Queue operations
  - **CreateQueue**
  - **DeleteQueue**
  - **Get/Set Metadata**
  - **Clear Messages**
- Message operations
  - **PutMessage** – Reads message and hides for time period
  - **GetMessages** – Reads one or more messages and hides them
  - **PeekMessages** – Reads one or more messages w/o hiding them
  - **DeleteMessage** – Permanently deletes messages from queue

# Queue's Reliable Delivery

- Guarantee delivery/processing of messages (two-step consumption)
  - Worker Dequeues message and it is marked as Invisible for a specified “Invisibility Time”
  - Worker Deletes message when finished processing
  - If Worker role crashes, message becomes visible for another Worker to process

# Removing Poison Messages

Producers



Consumers

$C_1$

1. GetMessage(Q, 30 s)  $\rightarrow$  msg 1

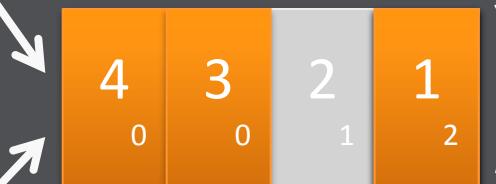
$C_2$

2. GetMessage(Q, 30 s)  $\rightarrow$  msg 2

# Removing Poison Messages

Producers

$P_2$



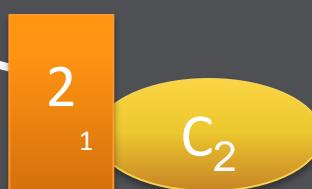
$P_1$

2. GetMessage(Q, 30 s) → msg 2
3. C<sub>2</sub> consumed msg 2
4. DeleteMessage(Q, msg 2)
7. GetMessage(Q, 30 s) → msg 1

Consumers



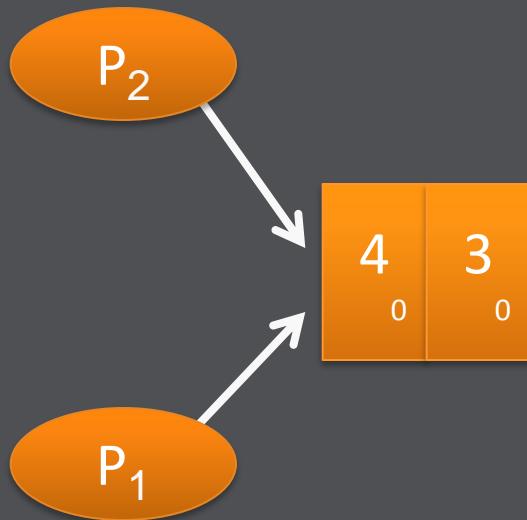
1. GetMessage(Q, 30 s) → msg 1
5.  $C_1$  crashed



6. msg1 visible 30 s after Dequeue

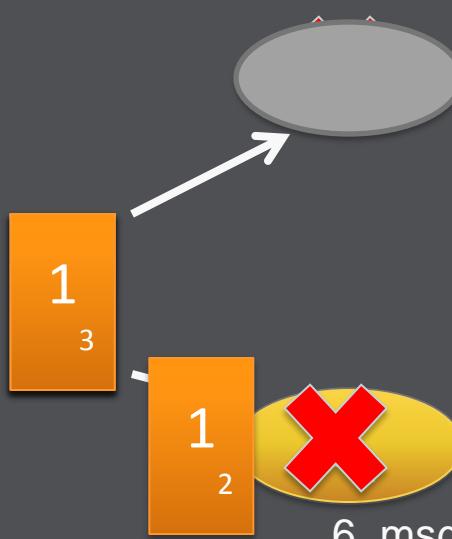
# Removing Poison Messages

Producers



2. Dequeue(Q, 30 sec) → msg 2
3. C2 consumed msg 2
4. Delete(Q, msg 2)
7. Dequeue(Q, 30 sec) → msg 1
8. C2 crashed

Consumers



1. Dequeue(Q, 30 sec) → msg 1
5.  $C_1$  crashed
10.  $C_1$  restarted
11. Dequeue(Q, 30 sec) → msg 1
12. DequeueCount > 2
13. Delete (Q, msg1)

6. msg1 visible 30s after Dequeue
9. msg1 visible 30s after Dequeue

# HowTo: Removing Poison Messages

```
while (true) {  
    // Retrieve message from queue  
    var msg = cloudStorage.OrderImportQueue.GetMessage(TimeSpan.FromSeconds(15));  
    if (msg != null) {  
        try {  
            // Message Processing  
            [...]  
        }  
        catch (Exception) {  
            if (msg.DequeueCount > 1) {  
                // Remove poisoned message  
                cloudStorage.OrderImportQueue.DeleteMessage(msg);  
            }  
        }  
    }  
    else {  
        Thread.Sleep(1000);  
    }  
}
```

# Queue Tips & Tricks

- Keep queue messages small
  - Use other storage mechanism for payload
- Use multiple queues (parallel) for high throughput
- Handle poisoned messages
- Only polling, no push mechanisms
  - If you need push use e.g. WCF instead
- Don't use queues for producer/consumer scenarios inside process
  - Use .NET 4 concurrent collections instead

# demo

## Queues

Windows Azure  
Queues for role  
communication

# Demo Content

- Queue management with Cerebrata
- If necessary
  - Speak about producer/consumer pattern
  - Concurrent queues in .NET
- Split existing web application into web/worker role

# Windows Azure Storage Summary

- Fundamental data abstractions to build your applications
  - Blobs – Files and large objects
  - Drives – NTFS APIs for migrating applications
  - Tables – Massively scalable structured storage
  - Queues – Reliable delivery of messages
- Easy to use via the Storage Client Library
- Hands on Labs

SQL Server in the Cloud

# SQL AZURE

# SQL Azure Database



## Managed Service

- Easy provisioning and deployment
- Auto high-availability and fault tolerance
- No need for server or VM administration



## Scale On Demand

- Database utility; pay as you grow
- Business-ready SLAs
- Enable multi-tenant solutions
- World-wide presence



## Innovate Faster

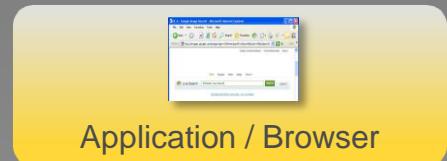
- Build cloud-based database solutions on consistent relational model
- Leverage existing skills through existing ecosystem of developer and management tools

# Comparing SQL Server and SQL Azure

- Goal is symmetry:
  - Database - T-SQL, features
  - Tools
  - Connectivity
  - Frameworks
- Some variations exist:
  - Table design
  - Some features
  - Scale strategy
  - Overview: <http://msdn.microsoft.com/en-us/library/ff394102.aspx>
- Differences are being reduced

# Application Topologies

From  
Windows Azure



Application / Browser



Windows  
Azure

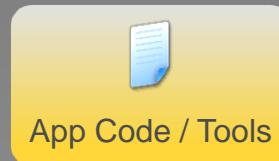
Microsoft  
Datacenter



SQL Azure

Code Near

From Outside  
Microsoft Datacenter



App Code / Tools



Microsoft  
Datacenter

SQL Azure

Code Far

From Windows Azure & Outside  
Microsoft Datacenter



App Code / Tools



SQL  
Server

SQL  
Azure  
Data  
Sync



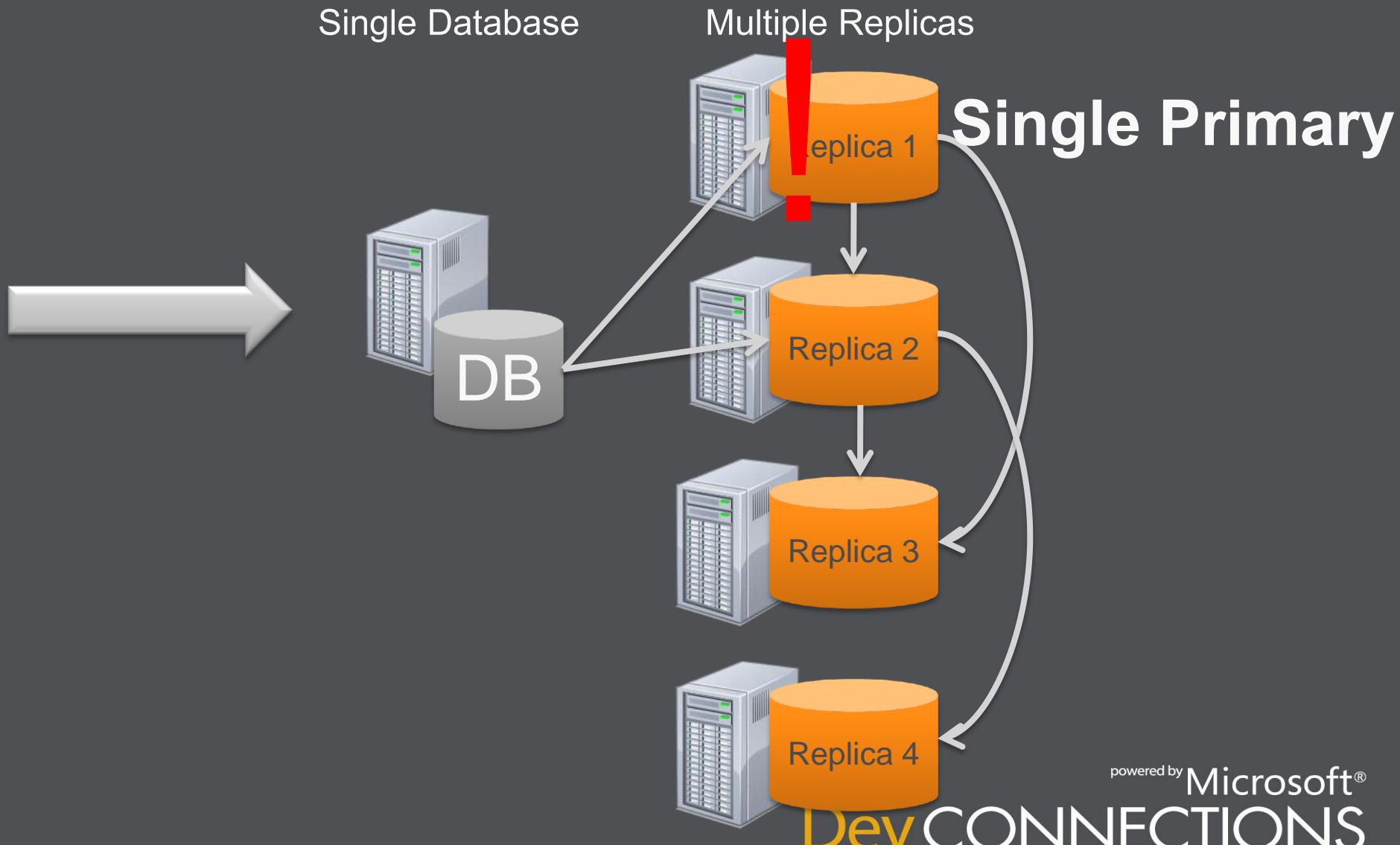
SQL Azure

Hybrid

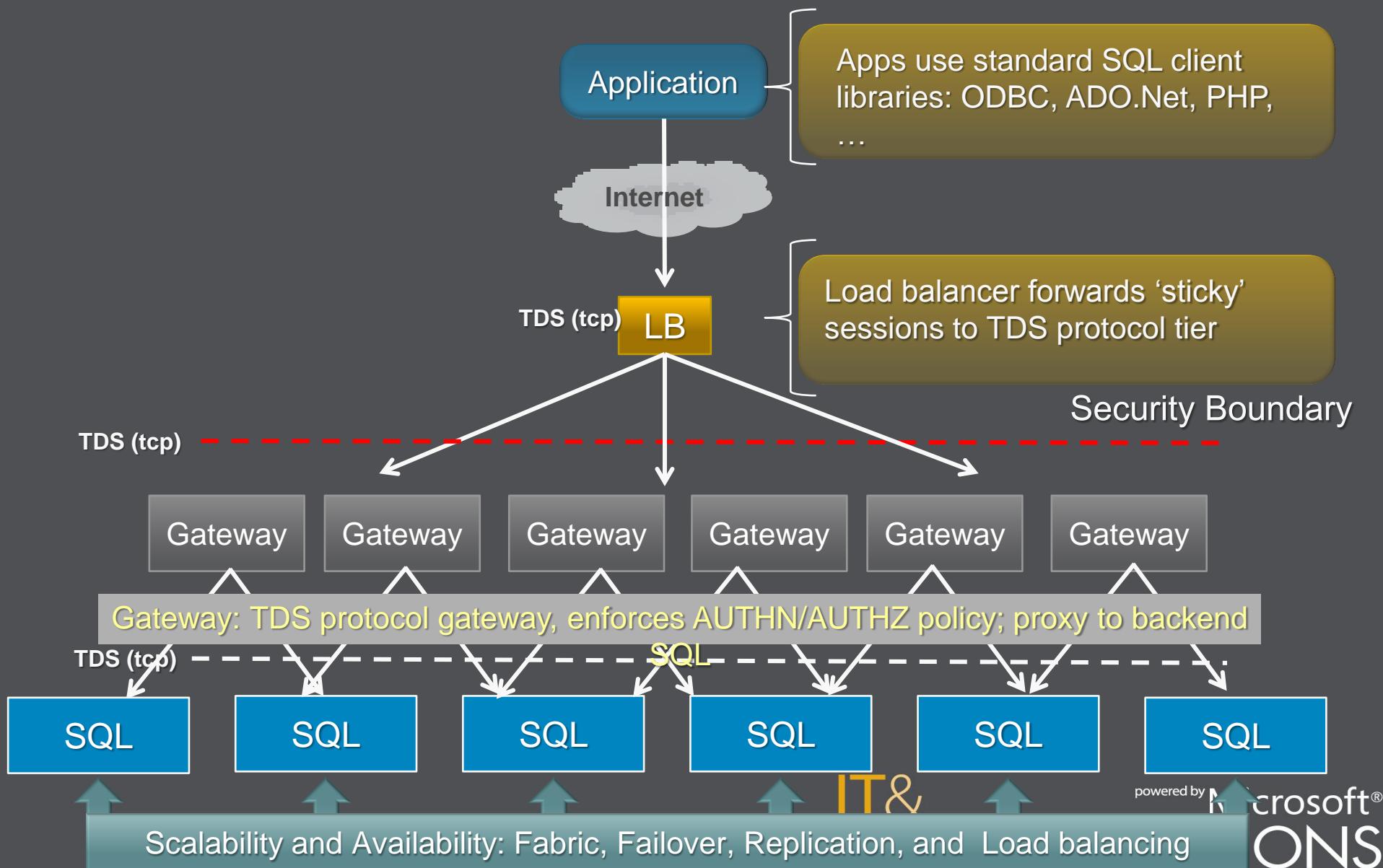
# Architecture

- Shared infrastructure at SQL database and below
  - Request routing, security and isolation
- Scalable technology provides the glue
  - Automatic replication and failover
- Provisioning, metering and billing infrastructure

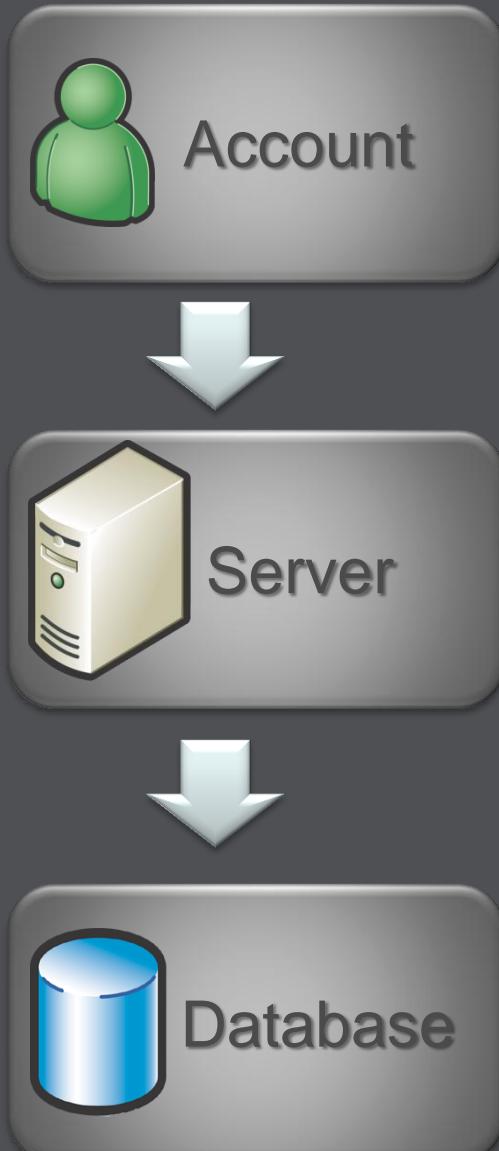
# Database Replicas



# Behind the Scenes of SQL Azure



# Service Provisioning Model



- Each **account** has zero or more **logical servers**
  - Provisioned via a common portal
  - Establishes a billing instrument
- Each **logical server** has one or more **databases**
  - Contains metadata about database & usage
  - Unit of authentication, geo-location, billing, reporting
  - Generated DNS-based name
- Each **database** has standard SQL objects
  - Users, Tables, Views, Indices, etc
  - Unit of consistency

# Connection Model

- SQL Azure exposes native SQL Server TDS protocol
- Use existing client libraries
  - ADO.NET, ODBC, PHP
  - OLE DB not officially supported
- Client libraries pre-installed in Windows Azure roles
- Support for ASP.NET controls
- Applications connect directly to a database
  - Cannot hop across DBs (no USE)

# Connecting to SQL Azure

- SQL Azure connection strings follow normal SQL syntax
- Applications connect directly to a database
  - “Initial Catalog = <db>” in connection string
  - No support for context switching (no USE <db>)
- Encryption security
  - Set **Encrypt = True**, only SSL connections are supported
  - **TrustServerCertificate = False**, avoid Man-In-The-Middle-Attack!
- Format of username for authentication:
  - ADO.Net:  
Data Source=server.database.windows.net;  
User ID=user@**server**;Password=password;...
- Setup your firewall rules first!

# Dealing with throttling

- Throttling provides a good experience to all databases on a single node
- SQL Azure is able to monitor and rebalance databases automatically
- Rebalancing uses a swap / move replica procedure
- Resource reservation is coming for a more predictable experience

# What causes throttling?

- Lock Consumption
  - Sessions using > 1M locks
- Excessive Log Usage
  - Single transaction consuming > 1GB
- Uncommitted Transactions
  - Single transaction consuming > 20% of the log
- Excessive TempDB usage
  - Session using > 5GB of TempDB
- Excessive Memory Usage
  - When the node experiences memory contention...
    - Sessions using > 16MB for > 20 seconds are terminated in descending order
- Idle Connections (30 mins)

# Logical vs. Physical Administration

- SQL Azure focus on logical administration
  - Schema creation and management
  - Query optimization
  - Security management (Logins, Users, Roles)
- Service handles physical management
  - Automatically provides HA “out of box”
  - Transparent failover in case of failure
  - Load balancing of data to ensure SLA

# Deployment

- Deploy via T-SQL scripts
- Support for SQL Server Data-Tier Applications (DAC) feature
  - DACPAC/BACPAC is unit of deployment
  - Cloud or on-premise is a deployment time choice
- Create Logical Server in same region as Windows Azure Affinity Group for code-near architecture
- Database Copy
  - Template databases

# Security Model

- Uses regular SQL security model
  - Authenticate logins, map to users and roles
  - Authorize users and roles to SQL objects
- Support for standard SQL Auth logins
  - Username + password
- No Windows Auth
  - If needed use Azure Connect
    - Domain-join roles
    - Use on-premise SQL Server

# demo

## SQL Azure

### Working with SQL Azure

# Demo Content

- Server and database creation in admin portal
- Server management in admin portal
- Management studio access
- Deployment
  - SQL Azure Migration Wizard
  - BACPAC Deployment
  - Database Copy
- Programmability
  - ADO.NET access to SQL Azure
  - Entity Framework access to SQL Azure

# SQL Azure Compatibility

## Currently Supported

- Tables, indexes and views
- Stored Procedures
- Triggers
- Constraints
- Table variables,  
session temp tables (#t)
- Spatial types, HierarchyId

## Not Currently Supported

- Data Types
  - XML, Sparse Columns, FileStream
- Partitions
- Full-text indexes
- SQL-CLR
- Tables without clustered indexes
  - Needed for replication
- Collations

# Database Editions

- Two SQL Azure Database SKUs: *Web & Business*
  - **Web Edition:** 1 GB @ \$9.99/month | 5 GB @ \$49.95/month
  - **Business Edition:** Up to 50 GB @ \$99.99/10 GB/month  
10 GB @ \$99.99 | 20 GB @ \$199.98 | 30 GB @ \$299.97 | 40 GB @ \$399.96 | 50 GB @ \$499.95
- You specify Web or Business Edition
  - **Web:** EDITION = web
  - **Business:** EDITION = business
- You specify MAXSIZE
  - **Web:** MAXSIZE = 1GB | 5GB
  - **Business:** MAXSIZE = 10GB | 20GB | 30GB | 40GB | 50GB
    - *This is the maximum size we will not let you grow beyond*
    - *You will only be charged for the actual peak size in any one day rounded up*
    - *For example, a 3.4 GB Web Edition will be charged 5GB rate.*

```
CREATE DATABASE foo1 (EDITION='business', MAXSIZE=50GB);
CREATE DATABASE foo2 (EDITION='business', MAXSIZE=30GB);
ALTER DATABASE foo2 MODIFY (EDITION='web', MAXSIZE=5GB);
```



**Business Edition**  
Up to 50 GB  
10 GB increments



**Web Edition**  
1 GB or 5 GB

# What's coming in Next Service Release?

- No-code OData endpoints
  - Already in [SQL Azure Labs](#)
- Point in Time Restore Preview
  - Restore a database to a specific point in time
  - Provides a 2 week window
- Support for Windows Azure Platform CoAdmin
- DB Import & Export in the Portal
- RePowering SQL Azure with SQL Server Denali Engine
- Sparse Columns
- Localized Portal and Engine error messages

# demo

## OData and SQL Azure

Public OData  
Endpoint for SQL  
Azure

# Demo Content

- Show OData configuration in SQL Azure Labs

Configure OData Service

Connection Information

Server Name:  Create a New Server

User Name:

Password:

Disconnect

Database Information

Database:

Enable OData

User Mapping

The OData interface to SQL Azure allows you to map both specific users to ACS access keys or to allow anonymous access through a single SQL Azure user. Whichever options you chose, please keep in mind that the interface will impersonate the SQL Azure user you choose. Please ensure you configure these user's security access accordingly.

Anonymous Access User:

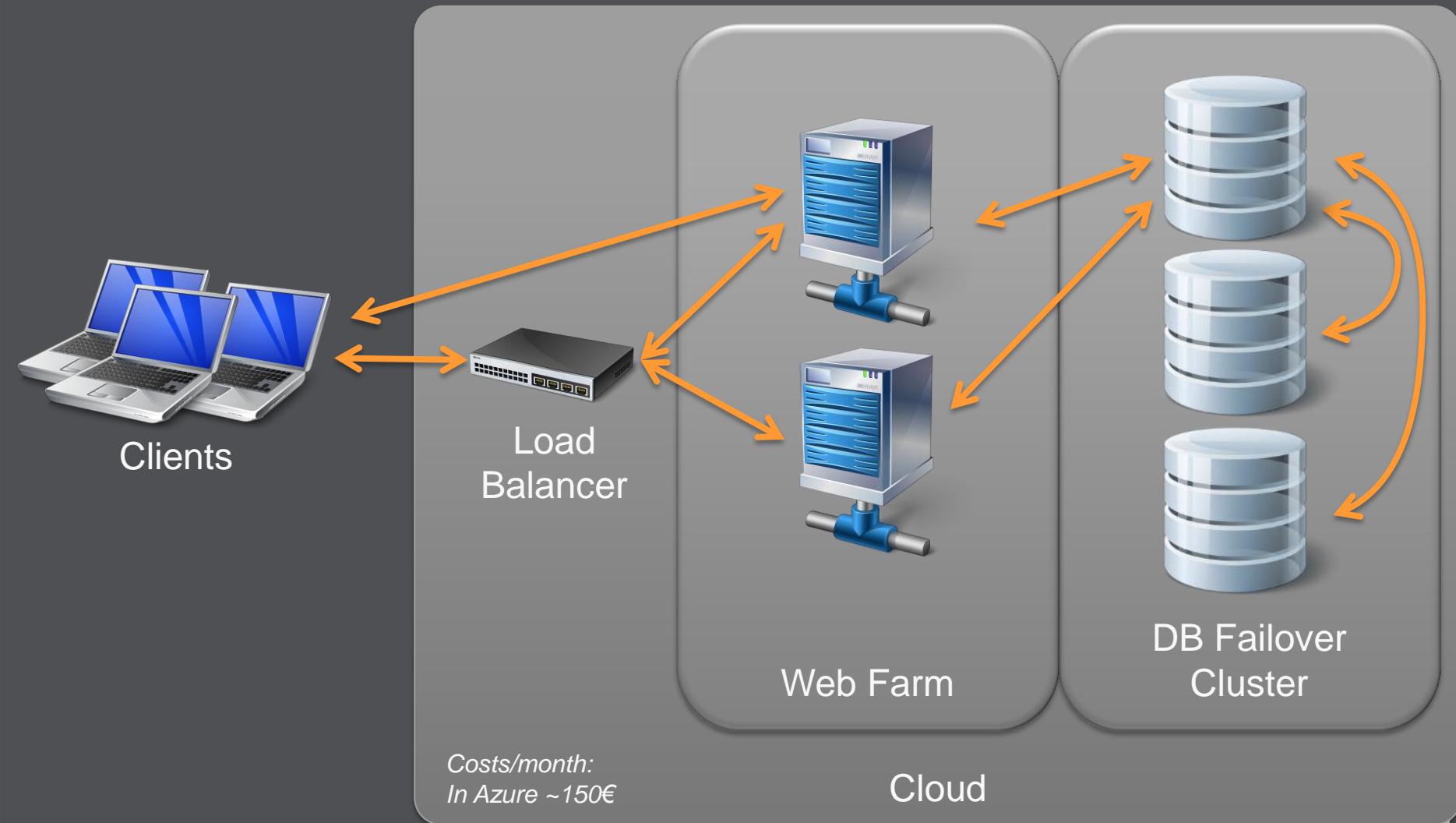
<https://odata.sqlazurelabs.com/OData.svc/v0.1/ygvyzufd89/AdventureWorksLight>

- Show queries to OData endpoint in IE

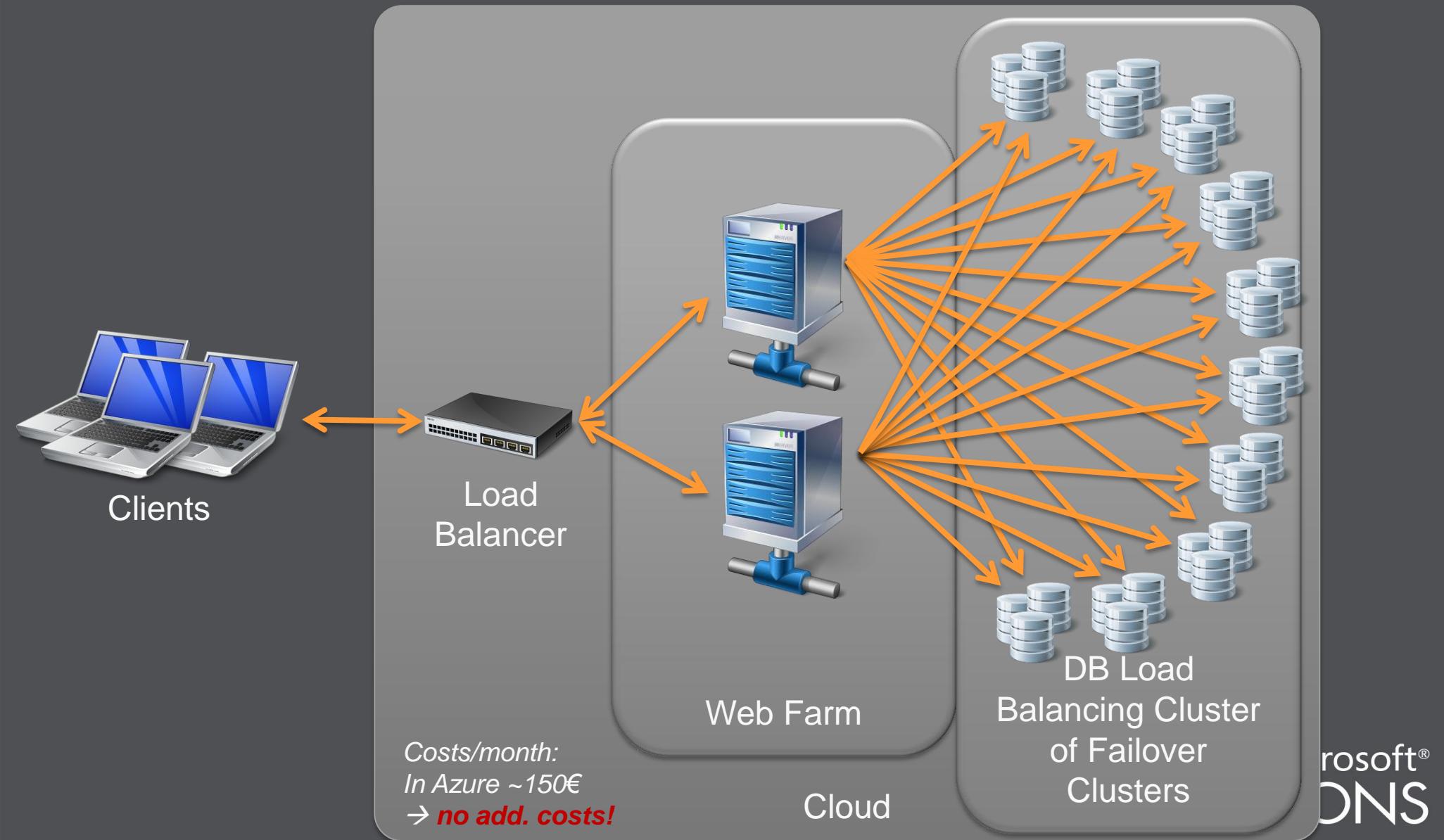
SQL Azure Future

# SCALING DATABASE APPLICATIONS

# Typical Architecture



# Solution = Sharding



# Scaling database applications

- Scale up
  - Buy large-enough server for the job
    - But big servers are expensive!
  - Try to load it as much as you can
    - But what if the load changes?
    - Provisioning for peaks is expensive!
- Scale-out
  - Partition data and load across many servers
    - Small servers are cheap! Scale linearly
  - Bring computational resources of many to bear
    - 800 little servers is very fast
  - Load spikes don't upset us
    - Load balancing across the entire data center

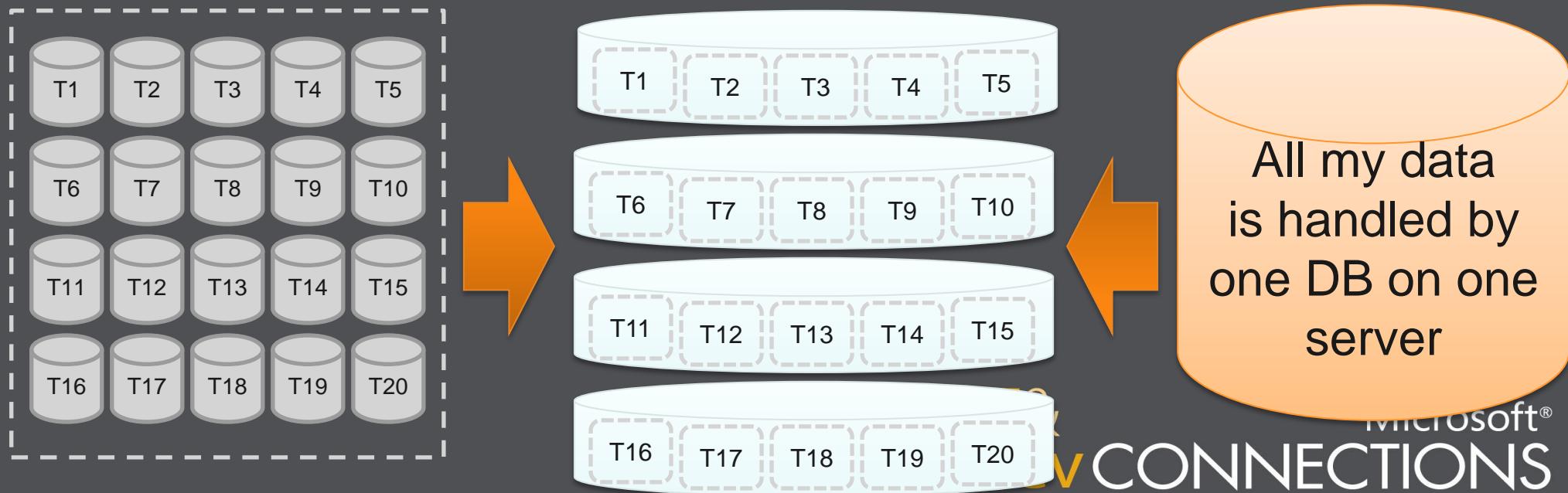


# Scale-out with SQL Azure Today

- Elastic **Provisioning** of Databases
  - CREATE DATABASE and go
  - No VMs, no servers
- **Pay-as-you-go** business model
  - Don't need it --- DROP it
- **Zero Physical Administration**
  - Built-in High Availability, patching, maintenance
- Database Copy, SQL Azure Data Sync

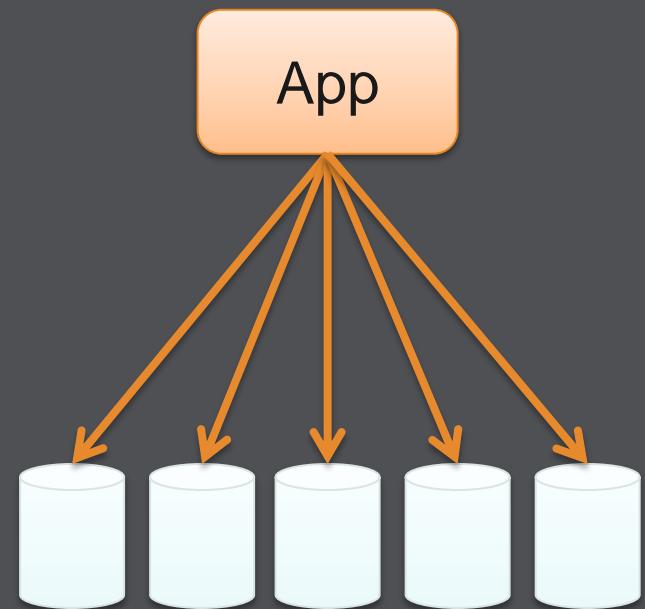
# Scale-out for Multi-tenant applications

- Put everything into one DB? Too big...
- Create a database per tenant? Not bad...
- Sharding Pattern: better
  - Application is already prepared for it!



# Sharding Pattern

- Linear scaling through database independence
  - No need for distributed transactions in common cases
- Engineered partitioning
  - Rather than complete transparency
- Local access for most
  - Connection routing
  - Query, transaction scoping
- Distributed access for some
  - Fan-out expensive computation



# Engineered Partitioning: Platform Capabilities

- Provisioning
  - Growing and shrinking capacity
- Managing
  - Upgrading, patching, HA for lots of databases

Covered by  
SQL Azure  
today

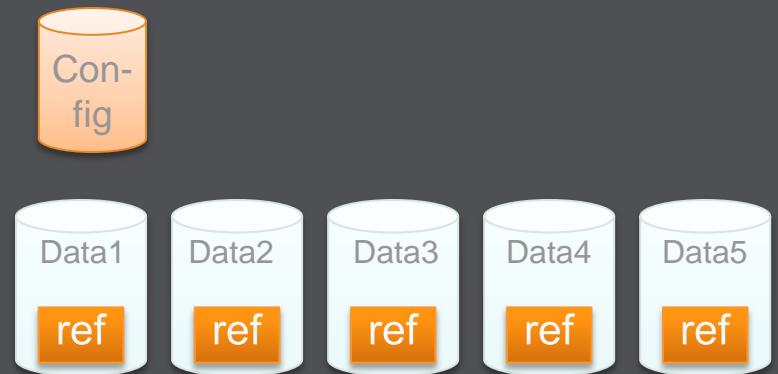
- Routing
  - Where is the directory?
  - How to scale it and use it?
- Partition Management
  - Splitting and Merging, without loss of availability
  - Fan-out

Coming up  
in SQL  
Azure:  
Federations

2011

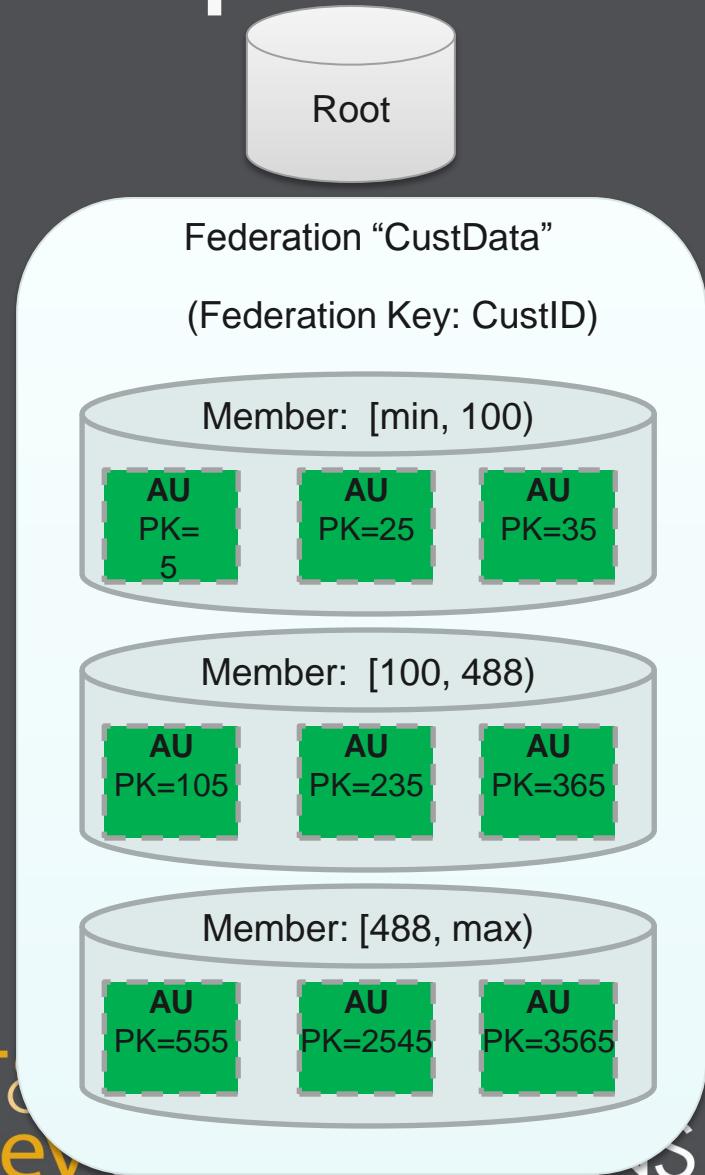
# Distribution of Data

- **Partitioned**
  - Spread across member machines
  - Each piece is on one machine (+HA)
  - Most of the data!
- **Centralized**
  - Only available in one place
  - Read and write, but not too much
- **Replicated**
  - Copied to all member machines
  - Can be read anywhere (reference)
  - Should not be written too much



# SQL Azure Federations: Concepts

- Federation
  - Represents the data being sharded
- Federation Key
  - The value that determines the routing of a piece of data
- Atomic Unit
  - All rows with the same federation key value: always together!
- Federation Member (aka Shard)
  - A physical container for a range of atomic units
- Federation Root
  - The database that houses federation directory

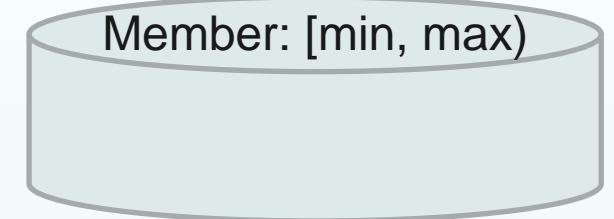


# Creating a Federation

- Create a root database
  - CREATE DATABASE SalesDB
  - Location of *partition map*
  - Houses *centralized* data
- Create the federation in root
  - CREATE FEDERATION Orders\_Fed (RANGE BIGINT)
  - Specify name, federation key type
    - Start with integral, guid types
  - Creates the first member, covering the entire range

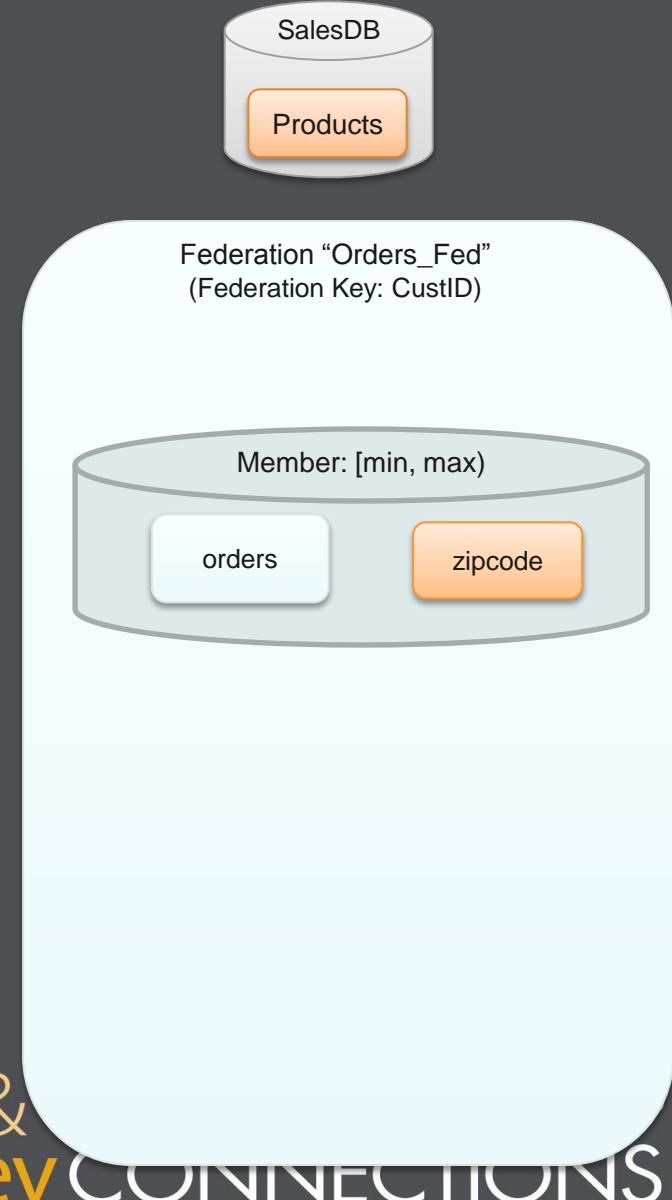


Federation “Orders\_Fed”  
(Federation Key: CustID)



# Creating the schema

- **Federated tables**
  - CREATE TABLE orders (...) FEDERATE ON (customerId)
  - Federation key must be in all unique indices
    - Part of the primary key
  - Value of federation key will determine the member
- **Reference tables**
  - CREATE TABLE zipcodes (...)
  - Absence of FEDERATE ON indicates reference
- **Centralized tables**
  - Create in root database



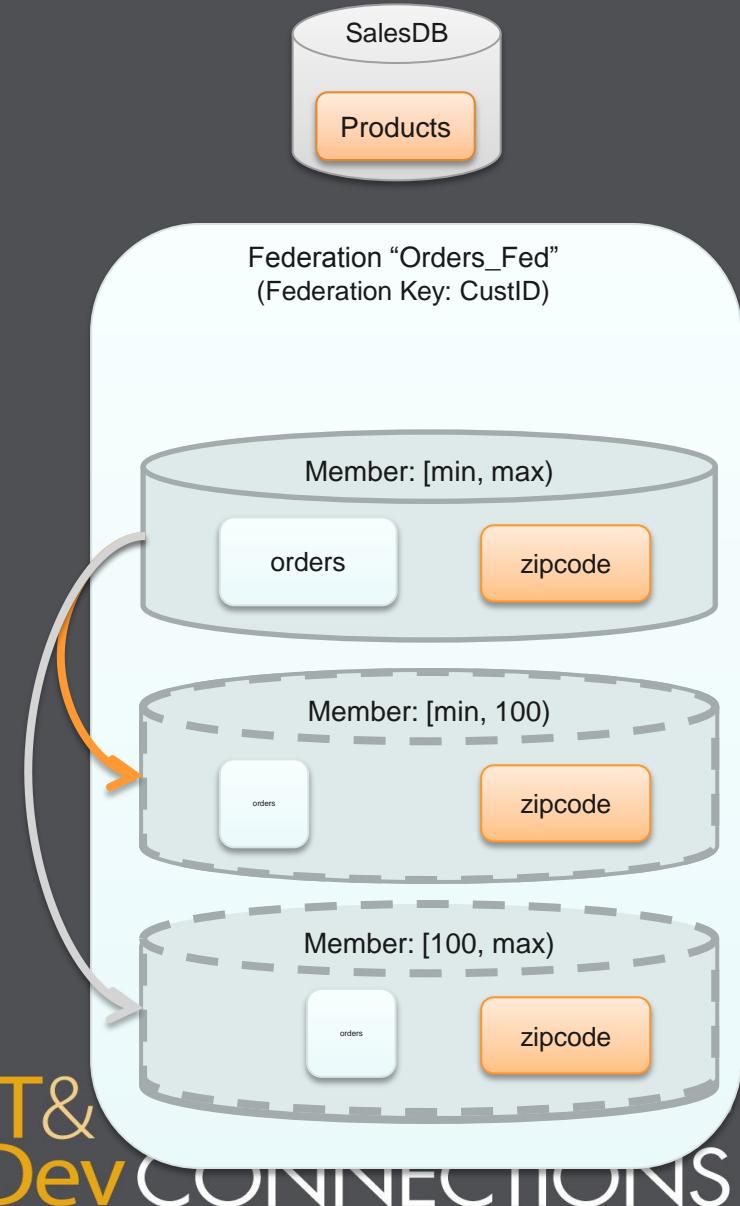
# Splitting and Merging

- **Splitting a member**

- When too big or too hot
- `ALTER FEDERATION Orders_Fed SPLIT (100)`
- Creates two new members
  - Splits (filtered copy) federated data
  - Copies reference data to both
- **Online!**

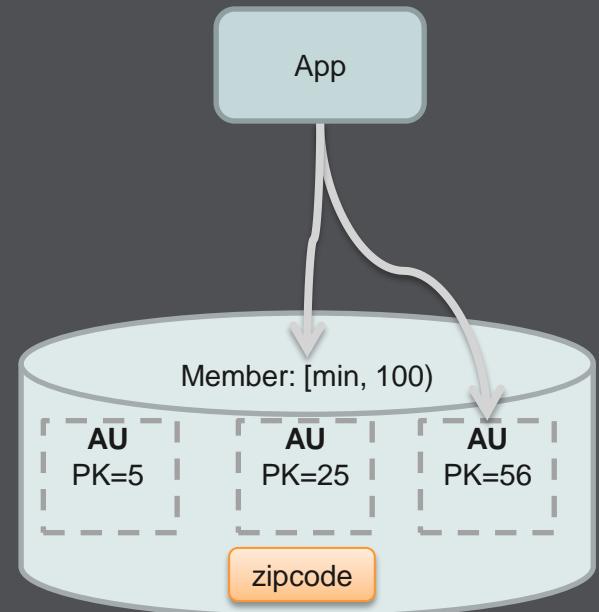
- **Merging members**

- When too small
- `ALTER FEDERATION Orders_Fed MERGE (200)`
- Creates new member, drops old ones



# Connecting and Operating

- Connect to atomic unit
  - USE FEDERATION Orders\_Fed (56) WITH FILTERING=ON
  - Connection routed to member containing 56
  - **Only data with federation key value 56 is visible**
    - Plus reference data
  - Safe: atomic unit can never be split
- Connect to entire federation member
  - USE FEDERATION Orders\_Fed (56) WITH FILTERING=OFF
  - Connection routed to member containing 56
  - **All data within the member database is visible**
  - Dangerous: federation member can be split



# Schema Distribution

- Federation members can have different schemas at a point in time:
  - Temporary, while schemas are being upgraded
  - Temporary, while customer is testing new schema on some shards
  - Permanently, because shards are different
- To alter schema:
  - Manually
    - Connect to each federation member (USE FEDERATION Orders(56) WITH FILTER=OFF)
    - Alter it (ALTER TABLE Customers ...)
  - Future: schema-distribution service
    - Connect to root
    - Manage and apply schemas asynchronously

# Sharding in SQL Azure: Beyond v1

- Schema Management
  - Allow multi version schema deployment and management across federation members.
- Fan-out Queries
  - Allow single query that can process results across large number of federation members.
- Auto Repartitioning
  - SQL Azure manages the federated databases for you through splits/merges based on some policy (query response time, db size etc)
- Multi Column Federation Keys
  - Federate on enterprise\_customer\_id+account\_id

# HowTo: Setup Federation

```
CREATE FEDERATION Orders_Fed (RANGE BIGINT)
```

```
USE FEDERATION Orders_Fed(0) WITH FILTERING=OFF
```

```
CREATE TABLE orders(orderidbigint, odatedatetime, customeridbigint,  
primary key (orderid, customerid))  
FEDERATE ON (customerid)
```

```
CREATE UNIQUE INDEX o_idx1 on orders(customerid, odate)  
CREATE INDEX o_idx2 on orders(odate)
```

```
CREATE TABLE orderdetails(orderdetailidbigint, orderidbigint, partidbigint, customeridbigint,  
primary key (orderdetailid, customerid))  
FEDERATE ON (customerid)
```

```
ALTER TABLE orderdetails add constraint orderdetails_fk1 foreign key(orderid,customerid)  
references orders(orderid,customerid)
```

```
CREATE TABLE uszipcodes(zipcodenvarchar(128) primary key, state nvarchar(128))
```

# HowTo: Work With Federation

```
Connect to: InitialCatalog='SalesDB'
```

- get some regular work done (within customer 239)

```
USE FEDERATION Orders_fed(239) WITH FILTERING=ON
```

```
SELECT * FROM Orders JOIN OrderDetailsON ...
```

```
INSERT INTO Orders (customerid, orderid, odate) VALUES (239, 2, '5/7/2010')
```

- get some cross-customer work done

```
USE FEDERATION Orders_fed(0) WITH FILTERING=OFF
```

```
DELETE from Orders WHERE odate < '1/1/2000'
```

- Repeat for other members...

- go back to root

```
USE FEDERATION ROOT
```

```
UPDATE CleanupSchedule set LastCleanupDate = GETSYSTIME()
```

# HowTo: Change Federation

--Day#2 business grows!

```
ALTER FEDERATION Orders_Fed SPLIT AT(1000)
```

--Day#3 black friday!

```
ALTER FEDERATION Orders_Fed SPLIT AT(100)
```

```
ALTER FEDERATION Orders_Fed SPLIT AT(200,300,400...)
```

--Day#4 recession hits!

```
ALTER FEDERATION Orders_Fed MERGE AT(100)
```

--Day#5 oh boy... double dip.

```
ALTER FEDERATION Orders_Fed MERGE AT(200,300,400...)
```

# Federation Technology Preview Nominations

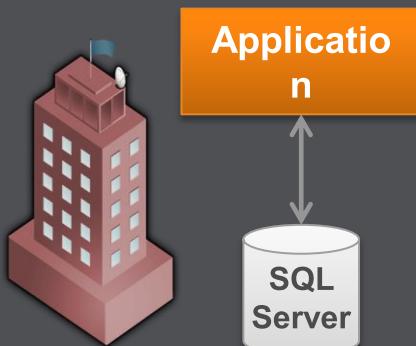
- **Now Open! Federations Technology Preview Program Nominations**
- Information on How to Nominate your Application
  - <http://blogs.msdn.com/cbiyikoglu/>
  - Click on the Survey Link
  - Fill-out the Survey Questions
  - Wait for communication from the technology preview team!

SQL Azure Future

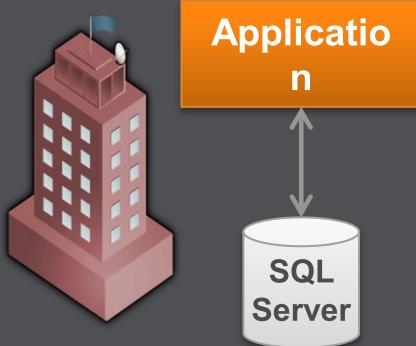
# SQL AZURE DATA SYNC

# Scenarios

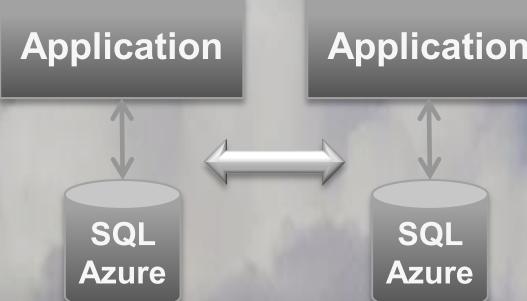
## On-Premises



E.g. Single location,  
branch office, retail  
E.g. One-way publish,  
two-way sharing,  
aggregation



## Cloud



E.g. Reporting  
E.g. Web site  
reference data



E.g. Geo-located web  
applications  
E.g. Traffic manager

# SQL Azure Data Sync – Key Features

- **Elastic Scale**
  - Service scales as resources requirements grow
- **No-Code Sync Configuration**
  - Easily define data to be synchronized
- **Schedule Sync**
  - Choose how often data is synchronized
- **Conflict Handling**
  - Handle issues where same data is changed in multiple locations
- **Logging and Monitoring**
  - Administration capabilities for tracking data and monitoring potential issues

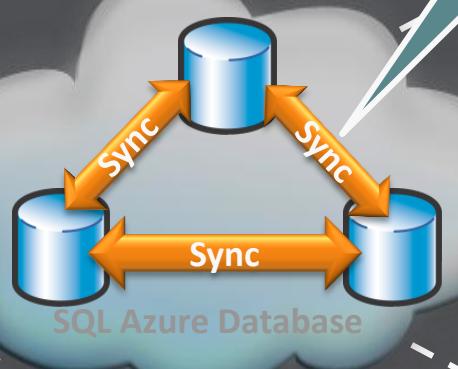
# Sync End to End Scenarios



On-Premises  
Applications



SQL Azure Data Sync  
CTP2  
(Coming in near future)



SQL Azure Data Sync CTP

Offline  
Applications



Retail &  
Remote  
Offices



Microsoft Sync Framework  
4.0  
CTP available now!

# V1 Planned UI

**Windows Azure Platform**

Billing | Nina Hu | Sign Out | ?

Save Discard Create Remove De-activate Add Remove from Sync Group Database Unregister

Manage Changes Sync Group

**Data Sync Service**

- Sync Groups**
  - Customer Data
  - HR Data
- Databases**
  - Cloud
    - Wirgccmqxs
    - Sales\_Cloud
    - HR\_Cloud
  - On-Premises
    - Agent\_NY
    - Sales\_Data

**Sync Group Name: Sales Data** ([Edit Name](#))      Status: **Activated**  
 Last synced at **12:00pm, Feb 24 2011** | Next sync in **21 minutes**

**Topology:**

**Table View**

**Configuration:**

- Conflict Resolution: **Last Writer Win**
- Sync Direction: **Bi-directional**
- Sync Schedule: **Every 30 Minutes**
- Sync Scope: [Edit Scope Setting](#)
- Synced Tables: **Customer**
- Synced Columns:
 

Column Name	Filter Value
ID	
Name	
State	WA
Address	
Phone	

[Take me back to the old portal](#) | © 2010 Microsoft Corporation [Privacy Statement](#) [Term of User](#) | [Feedback](#)

# SQL Azure Data Sync Roadmap

- Limited CTP2:
  - Available now, but closed
- Public CTP3:
  - Q3 2011
- V1:
  - Q4 2011

# Summary

SQL Azure provides a highly available cloud database service.

- Managed Service
- Scale On Demand
- Innovate Faster

# Your Feedback is Important

Please fill out a session evaluation form.

Thank you!