Azure Storage

Blob

- Data sharing: shared documents, pictures, video, music etc
- Big Data: store raw/logs and compute/map reduce over data
- Backup

```
CloudStorageAccount storageAccount = CreateStorageAccountFromConnectionString(CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create a blob client for interacting with the blob service.
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

// Create a container for organizing blobs within the storage account.
CloudBlobContainer container = blobClient.GetContainerReference("democontainerblockblob");
await container.CreateIfNotExistsAsync();

CloudBlockBlob blockBlob = container.GetBlockBlobReference(ImageToUpload);
await blockBlob.UploadFromFileAsync(ImageToUpload, FileMode.Open);
```

Tables

- Nosql key value store
- load balance
- batch operation

```
// Create the batch operation.
TableBatchOperation batchOperation = new TableBatchOperation();

// The following code  generates test data for use during the query samples.
for (int i = 0; i < 100; i++)
{
    batchOperation.InsertOrMerge(new CustomerEntity("Smith", string.Format("{0}", i.ToString("D4")))
        {
            Email = string.Format("{0}@contoso.com", i.ToString("D4")),
            PhoneNumber = string.Format("425-555-{0}", i.ToString("D4"))
        });
}

// Execute the batch operation.
IList<TableResult> results = await table.ExecuteBatchAsync(batchOperation);
```

Queues

Reliable messaging system

```
// Insert a message into the queue using the AddMessage method.
await queue.AddMessageAsync(new CloudQueueMessage("Hello World!"));

// Peek at the message in the front of a queue without removing it from the queue using PeekMessage (PeekMessages lets you peek >1 message)
CloudQueueMessage peekedMessage = await queue.PeekMessageAsync();
if (peekedMessage != null)
{
    Console.WriteLine("The peeked message is: {0}", peekedMessage.AsString);
}

CloudQueueMessage message = await queue.GetMessageAsync();
if (message != null)
{
    Console.WriteLine("Processing & deleting message with content: {0}", message.AsString);
    await queue.DeleteMessageAsync(message);
}
```

Disk

Persistent of VMs in Azure

```
// Create a page blob in the newly created container.
CloudPageBlob pageBlob = container.GetPageBlobReference(PageBlobName);
await pageBlob.CreateAsync(512 * 2 /*size*/); // size needs to be multiple of 512 bytes

// Write to a page blob
byte[] samplePagedata = new byte[512];
Random random = new Random();
random.NextBytes(samplePagedata);
await pageBlob.UploadFromByteArrayAsync(samplePagedata, 0, samplePagedata.Length);

// Read from a page blob
int bytesRead = await pageBlob.DownloadRangeToByteArrayAsync(samplePagedata, 0, 0, samplePagedata.Count());
```

Azure Storage

- Disk Storage: SSD for VMs
- Object Storage: Object Storage, unstructured data
- File Storage: File share
- Data transport: move data to cloud
- hybird storage: cache and other storage

Every microsoft app have data stored in blob storage.

Azure Storage Architecture

Stream layer:

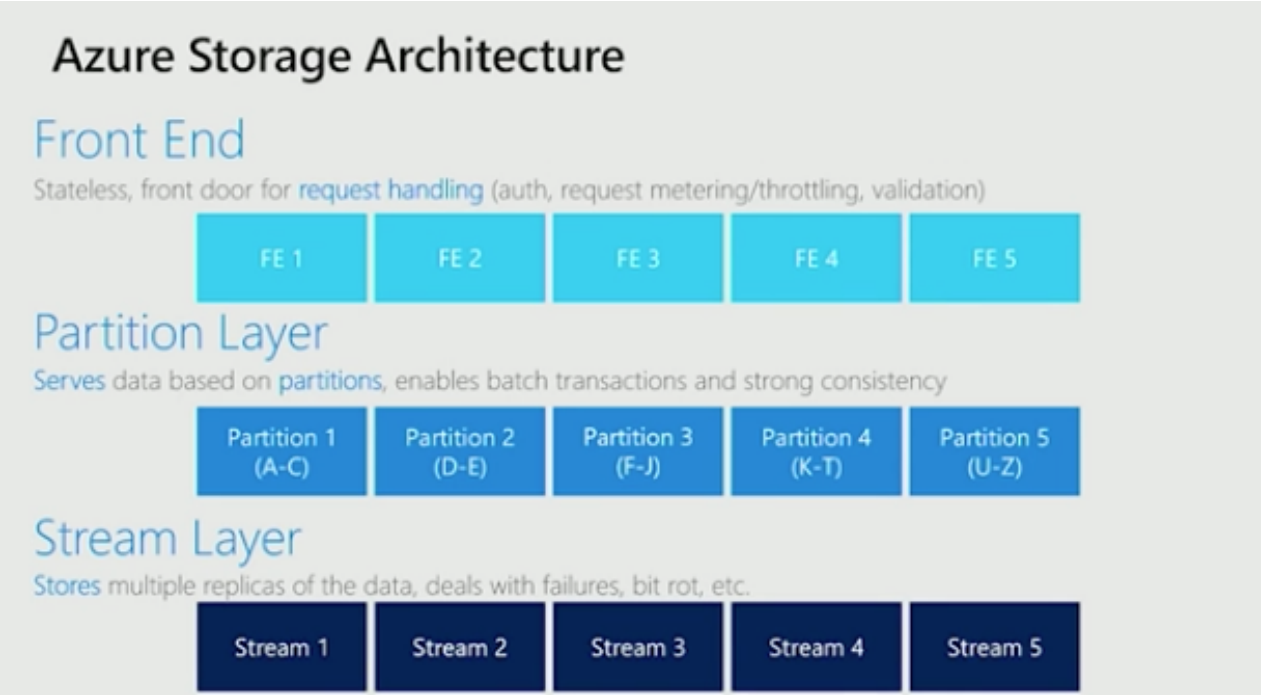
- Strong consistency
- monitor machine (disk, blob and file storage all on this layer)
- power layer
- the layer that contact the machine

Partition layer:

- Build the concept of blobs
- realizes that there is a lot of queries for this part of data, use more servers
- how to index, cache ur data
- load balance

Front End layer:

- authentication
- Restful API



Blob Storage

- Durable and Available

  replication option (not so clear about it)
- Secure: user own their key, data encrypted with that key
- Scalability
  - Change capacity and performance
  - increase throughput (more than 100G/s)
  - fact acess time
  - cost efficient

Instanenous enhancement

lower latency, consistent low latency

Soft Delete

create a snapshot automatically

Storage Metrics