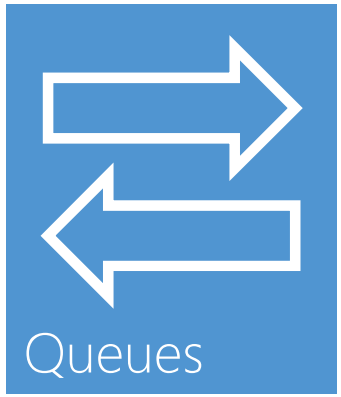
Azure Storage and Cognitive Services

Teerachai Laothong

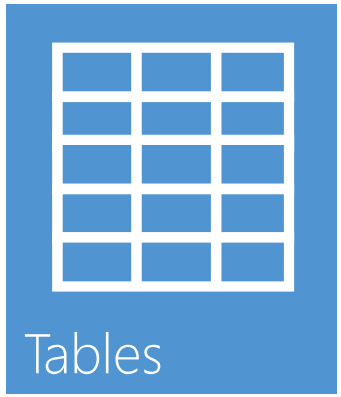
IMC Institute

Azure Storage

Storage for any type of data, analogous to files in a file system, with individual blobs storing up to 1 TB of data



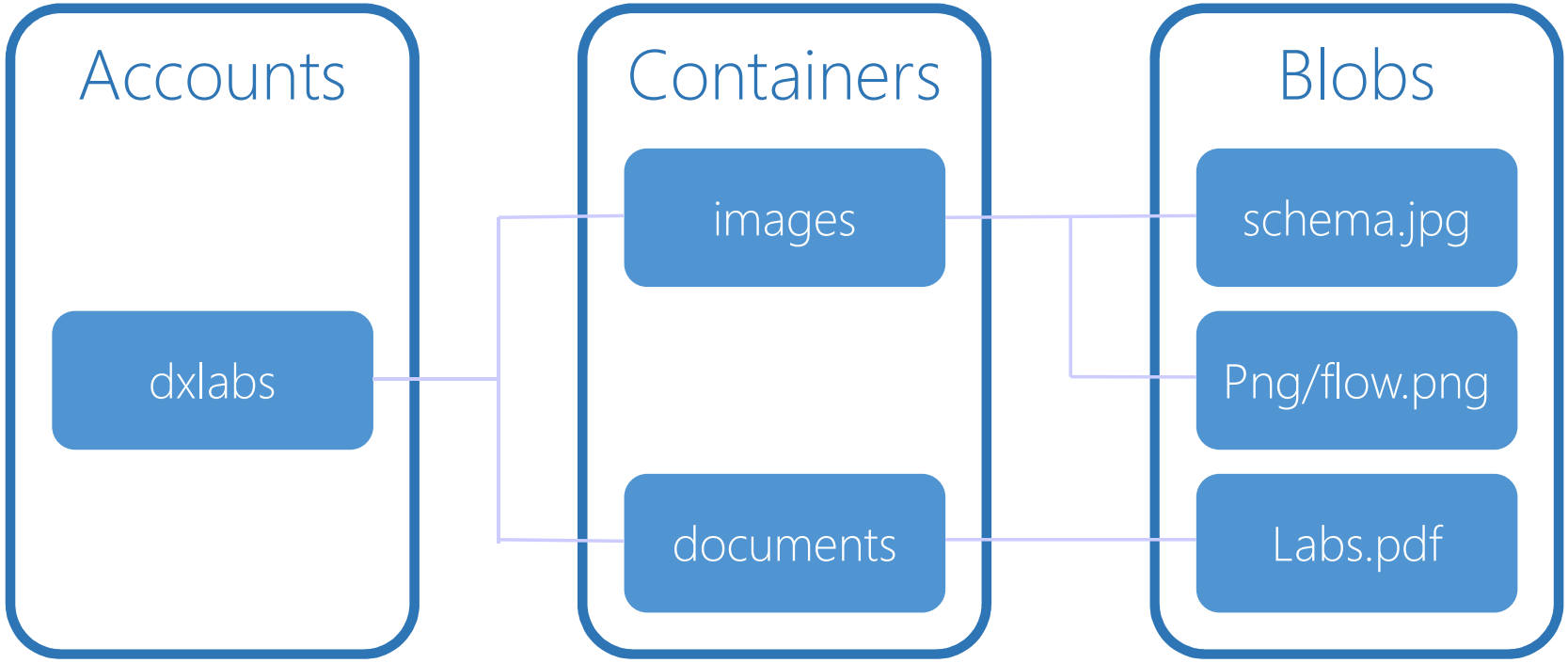
Reliable messaging for workflow processing and for communication between applications or application components



NoSQL data storage rapid development and fast access to large quantities of data

File sharing using Server Message Block (SMB) protocol

Blob Storage

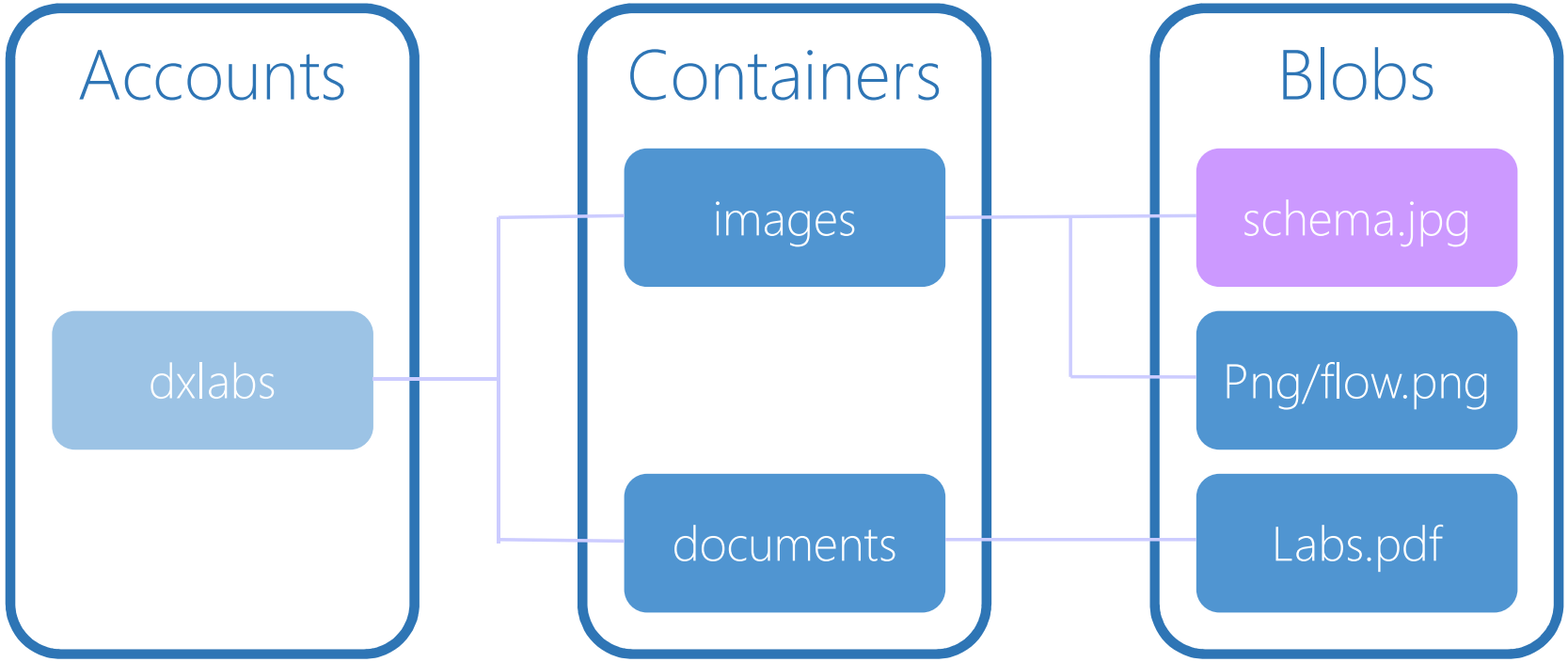


3 to 24 characters
0-9 and a-z
Unique within Azure

3 to 63 characters
0-9, a-z, and dashes

1 to 1,024 characters
Any characters (including slashes)
URL characters must be escaped
Max. 254 path segments

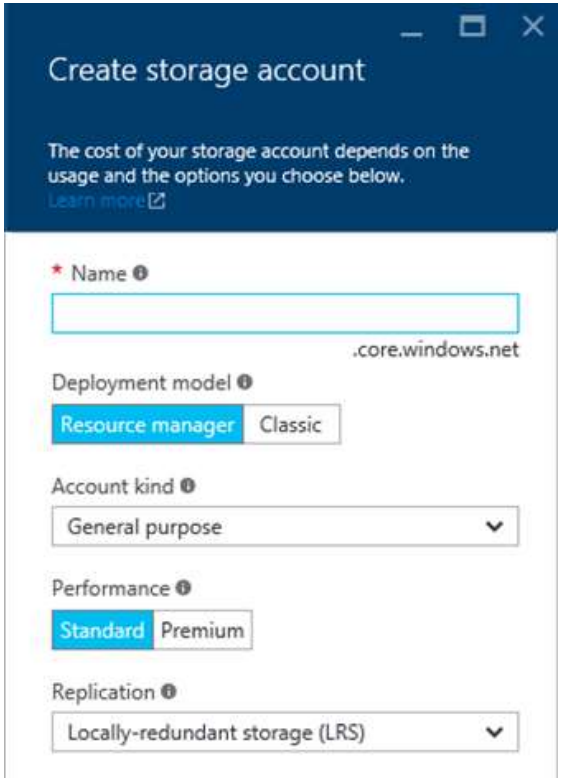
Blob URLs



https://dxlabs.blob.core.windows.net/images/schema.jpg

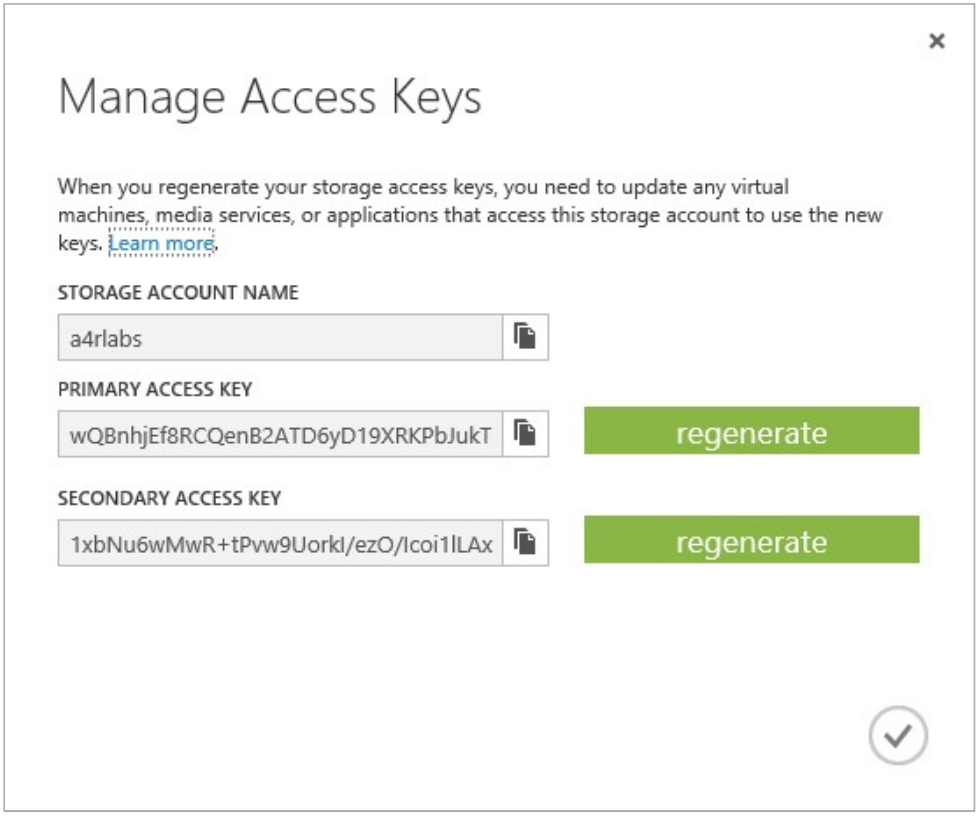
Storage Accounts

- Up to 500 TB of data per account
- Maximum of 100 storage accounts per subscription
- Two types of accounts
  - "General purpose" and "Blob storage"
- Four types of replication
  - LRS, ZRS, GRS, and RA-GRS
- Support optional 256-bit AES encryption (currently in preview)



Storage Keys

- Access to storage by non-account-owners relies on keys for authentication
  - Two 512-bit keys per account
- Keys should be "rolled" periodically for security
- Keys can be used to generate shared-access signatures (SAS) for secure and restricted access



Shared-Access Signatures

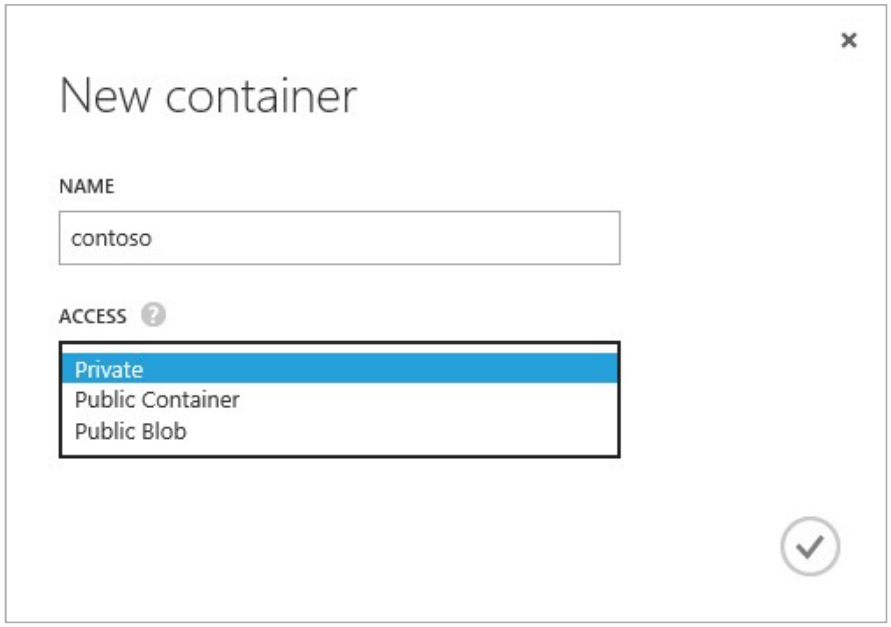
Blob URL

https://a4rlabs.blob.core.windows.net/images/schema.jpg?st=2016-02-07T19%3A58%3A00Z&se=2016-02-08T19%3A58%3A00Z&sp=r&sv=2015-02-21&sr=b&sig=BGebg1eduvPTwQnZWZlBphM8YGP9sRYt2WiPIL70vcw%3D

Query string containing shared-access signature

Storage Containers

- Unlimited number of blob containers per storage account
- Three access policies
  - Private – Blobs can't be read or enumerated anonymously
  - Public Container – Blobs can be read and enumerated anonymously
  - Public Blob – Blobs can be read anonymously, but cannot be enumerated



Storage Blobs

- Unlimited number of blobs per container
- Three types of blobs

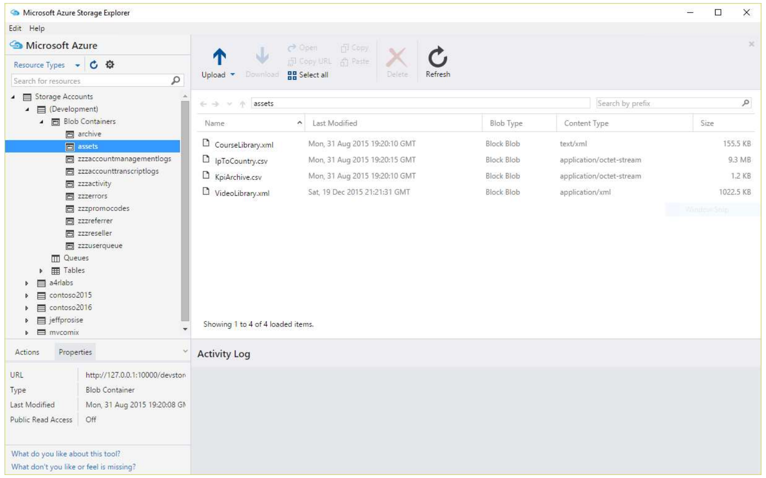
Block	Append	Page
Up to 195 GB	Up to 195 GB	Up to 1 TB
General-purpose streaming and storage	Optimized for append operations	VHDs only; optimized for random access

- Blobs also support user-defined metadata (key-value pairs)

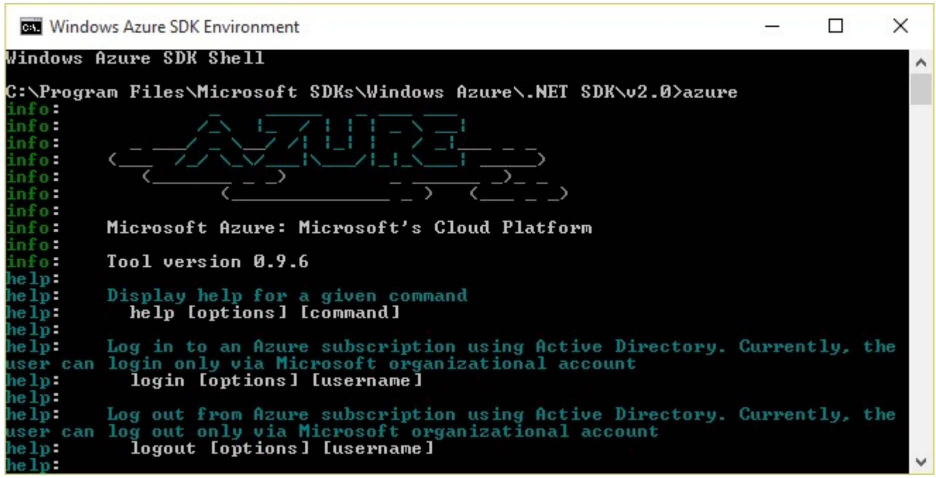
Azure Storage Tools

- Portal doesn't provide functionality for uploading blobs
- Use free, third-party, cross-platform tools instead

Microsoft Azure Storage Explorer



Azure Command-Line Interface (CLI)



Accessing Blob Storage Programmatically

- Blob service can be accessed using REST APIs
  - Accessible to any programming language that supports HTTP(S)
- Blob service can also be accessed using Azure Storage SDKs available for popular languages and platforms

- Also available from NuGet, NPM, and other package managers

Uploading a Blob (C#)

- Create a blob in the specified storage account and specified container using the Azure Storage SDK for .NET
- Upload the contents of a local file to the blob
- Get the connection string for the storage account from the Azure portal

```
CloudStorageAccount account =
    CloudStorageAccount.Parse("connection_string);
CloudBlobClient client = account.CreateCloudBlobClient();
CloudBlobContainer container =
    client.GetContainerReference("container_name");
CloudBlockBlob blob =
    container.GetBlockBlobReference("blob_name"));
await blob.UploadFromFileAsync("file_name");

// Or use UploadFromStreamAsync or
// UploadFromByteArrayAsync
```

Downloading a Blob (Node.js)

- Get a reference to a specified blob in a specified container in a specified storage account
- Download the blob and store its contents in a local file

```
var storage = require("azure-storage");
var service =
    storage.createBlobService("connection_string");
service.getBlobToLocalFile(
    "container_name", "blob_name", "file_name",
 function(error, result, response) {
    if (!error) {
        // File downloaded
    }
});

// Or use getBlobToStream, getBlobToTest, or
// createReadStream
```

Enumerating Blobs in a Container (C#)

- Enumerate all the block blobs in a specified container in a specified storage account
- Retrieve the name of each blob
- IListBlobItem could CloudBlockBlob, Cloud-PageBlob, or Cloud-AppendBlob

```
CloudStorageAccount account =
    CloudStorageAccount.Parse("connection_string);
CloudBlobClient client = account.CreateCloudBlobClient();
CloudBlobContainer container =
    client.GetContainerReference("container_name");

foreach (IListBlobItem item in container.ListBlobs())
{
    var blob = item as CloudBlockBlob;
    if (blob != null)
    {
        string name = blob.Name;
    }
}
```

Writing Blob Metadata (Node.js)

- Add metadata properties named "Property1," "Property2," and "Property3" to a blob

```
var storage = require("azure-storage");
var service =
    storage.createBlobService("connection_string");

var metadata = {
    "Property1", "Value1",
    "Property1", "Value2",
    "Property1", "Value3"
};

service.setBlobMetaData("container_name", "blob_name",
    metadata, function(error, result, response) {
    if (!error) {
        // Succeeded
    }
});
```

Reading Blob Metadata (C#)

- Read metadata properties named "Property1," "Property2," and "Property3" from a blob

```
blob.FetchAttributes();
string p1 = blob.Metadata.ContainsKey("Property1") ?
    blob.Metadata["Property1"] : null;
string p2 = blob.Metadata.ContainsKey("Property2") ?
    blob.Metadata["Property2"] : null;
string p3 = blob.Metadata.ContainsKey("Property3") ?
    blob.Metadata["Property3"] : null;
```

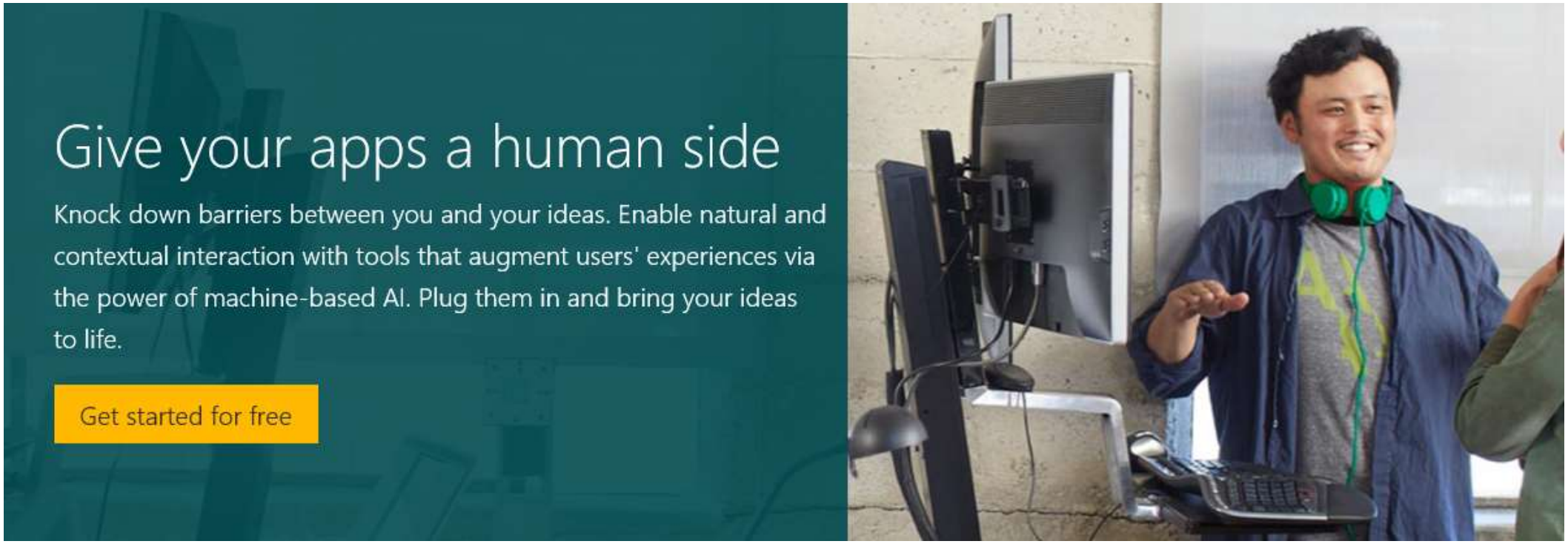
Deleting a Blob (Node.js)

- Get a reference to a specified blob in a specified container in a specified storage account
- Delete the blob

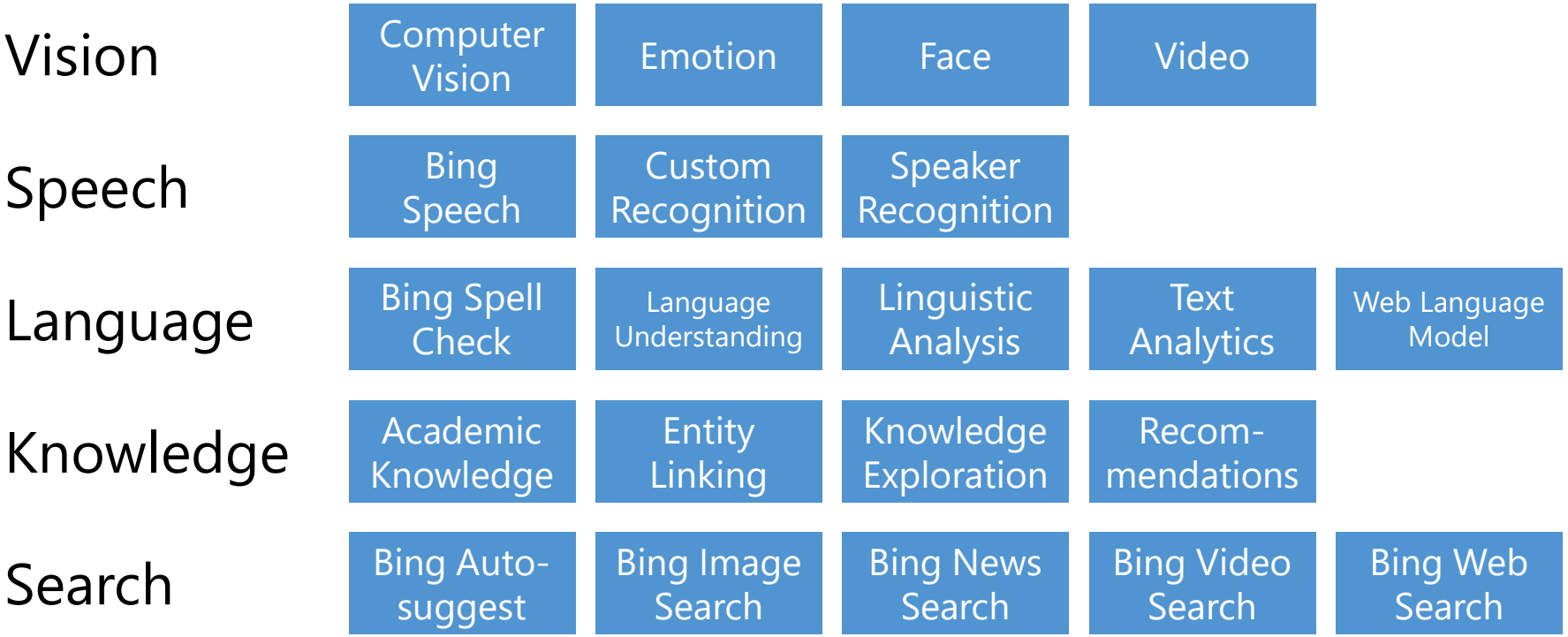
```
var storage = require("azure-storage");
var service =
    storage.createBlobService("connection_string");
service.deleteBlob("container_name", "blob_name",
    function(error, response) {
    if (!error) {
        // Blob deleted
    }
});
```

Microsoft Cognitive Services

- Intelligence APIs for building intelligent apps



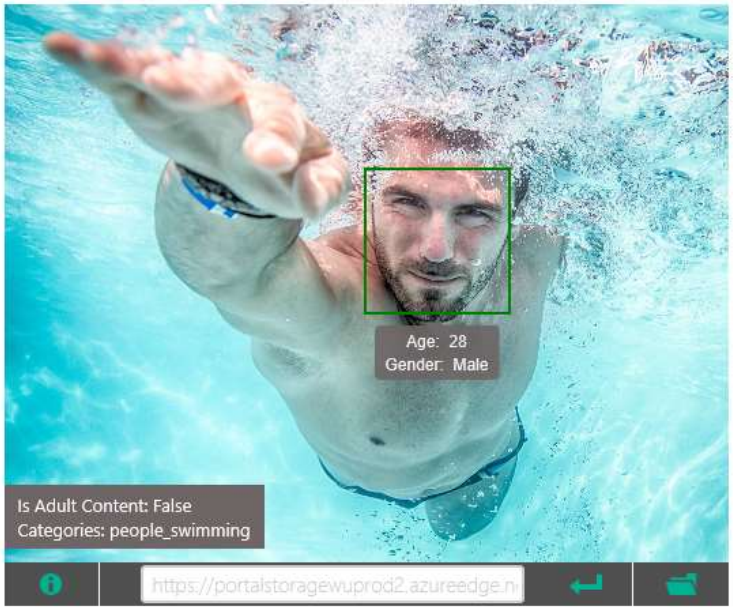
Cognitive Services APIs



Computer Vision API

Analyze an image

This feature returns information about visual content found in an image. Use tagging, descriptions and domain-specific models to identify content and label it with confidence. Apply the adult/racy settings to enable automated restriction of adult content. Identify image types and color schemes in pictures.



Features:

Feature Name	Value
Description	{ "type": 0, "captions": [{ "text": "a man swimming in a pool of water", "confidence": 0.7850108693093019 }] }
Tags	[{ "name": "water", "confidence": 0.9996442794799805 }, { "name": "sport", "confidence": 0.9504992365837097 }, { "name": "swimming", "confidence": 0.9062818288803101, "hint": "sport" }, { "name": "pool", "confidence": 0.8787588477134705 }, { "name": "water sport", "confidence": 0.631849467754364, "hint": "sport" }]
Image Format	jpeg
Image Dimensions	1500 x 1155
Clip Art Type	0 Non-clipart
Line Drawing Type	0 Non-LineDrawing
Black & White Image	False

Using the Computer Vision API (C#)

- Submit an image via URI to the Computer Vision API and ask for captions and descriptive tags
  - Optionally pass a stream instead of a URI
- Uses Microsoft.Project-Oxford.Vision NuGet package
- Other VisualFeatures include Adult, Category, Color, Faces, ImageType, and Tags

```
VisionServiceClient vision =
    new VisionServiceClient("subscription_key");
VisualFeature[] features =
    new VisualFeature[] { VisualFeature.Description };
AnalysisResult result =
    await vision.AnalyzeImageAsync(uri, features);

string caption = result.Description.Captions[0].Text);

foreach (string tag in result.Description.Tags)
{
    // tag holds descriptive tag for image (e.g., "river")
}
```

Using the Computer Vision API (Node.js)

- Submit an image via URI to the Computer Vision API and ask for captions and descriptive tags
  - Optionally pass a stream instead of a URI
- Other VisualFeatures include Adult, Category, Color, Faces, ImageType, and Tags

```
var options = {
    url: "https://api.projectoxford.ai/vision/v1.0/analyze",
    qs: { visualFeatures: "Description" },
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'Ocp-Apim-Subscription-Key': 'subscription_key'
    },
  ...
};
request(options, function(err, response, result) {
    if(!err) {
        var caption = result.description.captions[0].text;
    }
});
```

Hands-On Lab

Azure Storage and Cognitive Services HOL (MVC).html

Azure Storage and Cognitive Services HOL (Node).html