Certainly! Below is a complete architecture summary for processing images with handwritten text in Azure, using Cosmos DB for storing the extracted text.

### Architecture Components

1. \*\*Azure Blob Storage\*\*:

- \*\*Containers\*\*:

- `incoming-images`: Stores newly uploaded images.

- `processed-images`: Stores images after processing.

2. \*\*Azure Service Bus\*\*:

- \*\*Queue\*\*: `image-processing-queue` for managing image processing tasks.

- \*\*Dead-Letter Queue\*\*: Automatically handles messages that fail processing after retries.

3. \*\*Azure Functions\*\*:

- \*\*Service Bus Trigger\*\*: Processes messages from the `image-processing-queue`.

4. \*\*Azure Cognitive Services\*\*:

- \*\*Computer Vision API\*\* or \*\*Form Recognizer\*\* for OCR processing.

5. \*\*Azure Cosmos DB\*\*:

- Stores the extracted text and metadata.

### Detailed Workflow

1. \*\*Image Upload\*\*:

- User or system uploads an image to the `incoming-images` container in Azure Blob Storage.

- A message with the image name and language is placed in the `image-processing-queue`.

2. \*\*Azure Function (Service Bus Trigger)\*\*:

- \*\*Retrieve Image\*\*: Fetch the image from the `incoming-images` container.

- \*\*Perform OCR\*\*: Use Azure Cognitive Services to extract text from the image.

- \*\*Store Results\*\*: Save the extracted text and metadata to Cosmos DB.

- \*\*Move Image\*\*: Move the image to the `processed-images` container.

- \*\*Delete Original\*\*: Delete the original image from `incoming-images`.

3. \*\*Dead-Letter Queue\*\*:

- Messages that fail processing after retries are moved to the dead-letter queue for further inspection and handling.

### High-Level Diagram

```plaintext

+--------------------+ +--------------------+ +--------------------+

| Blob Storage | | Service Bus | | Azure Function |

| (incoming-images) | | (Queue) | | (Queue Trigger) |

+--------------------+ +--------------------+ +--------------------+

| | |

| v v

| +-----------------------+ +------------------------+

| | image-processing | | Fetch Image from Blob |

| | -queue- | | (incoming-images) |

| +-----------------------+ +------------------------+

| | |

| v v

| +-----------------------+ +------------------------+

| | Message contains | | Perform OCR using |

| | Image Name & | | Azure Cognitive |

| | Language | | Services |

| +-----------------------+ +------------------------+

| | |

v v v

+--------------------+ +-----------------------+ +------------------------+

| Blob Storage | | Azure Cosmos DB | | Move Image to |

| (processed-images) | | | | processed-images |

+--------------------+ +-----------------------+ +------------------------+

| |

v v

+--------------------+ +-----------------------+

| Store Extracted | | Delete Image from |

| Text and Metadata | | incoming-images |

+--------------------+ +-----------------------+

```

### Configuration and Best Practices

1. \*\*Azure Blob Storage\*\*:

- \*\*Access Control\*\*: Use Azure Active Directory (AAD) or Shared Access Signatures (SAS) for secure access to blob storage.

- \*\*Event Grid\*\*: Optionally, use Event Grid for event-driven architecture and trigger Azure Functions directly from blob storage events.

2. \*\*Azure Service Bus\*\*:

- \*\*Queue Setup\*\*: Configure the queue with dead-lettering enabled.

- \*\*Message TTL\*\*: Set appropriate message time-to-live (TTL) and maximum delivery count.

3. \*\*Azure Functions\*\*:

- \*\*Function Code\*\*: Implement the function to handle message processing, OCR, and data storage.

- \*\*Retry Policy\*\*: Define a robust retry policy to handle transient errors.

- \*\*Monitoring\*\*: Use Application Insights for monitoring and diagnostics.

4. \*\*Azure Cognitive Services\*\*:

- \*\*API Selection\*\*: Choose between Computer Vision API and Form Recognizer based on the complexity of the OCR tasks.

- \*\*API Keys\*\*: Securely store and manage API keys using Azure Key Vault.

5. \*\*Azure Cosmos DB\*\*:

- \*\*Database and Container Setup\*\*: Create a Cosmos DB account, database, and container to store extracted text and metadata.

- \*\*Partition Key\*\*: Choose an appropriate partition key for optimal performance and scalability.

- \*\*Indexing\*\*: Configure indexing policies based on query patterns.

- \*\*Throughput\*\*: Provision throughput (RU/s) based on expected load and adjust dynamically as needed.

### Azure Function Sample Configuration

\*\*Function Code (C# Example)\*\*:

```csharp

using System;

using System.Threading.Tasks;

using Microsoft.Azure.ServiceBus;

using Microsoft.Azure.WebJobs;

using Microsoft.Extensions.Logging;

using Azure.Storage.Blobs;

using Azure.AI.FormRecognizer;

using Azure.AI.FormRecognizer.DocumentAnalysis;

using Microsoft.Azure.Cosmos;

public static class QueueTriggerFunction

{

private static readonly string BlobConnectionString = Environment.GetEnvironmentVariable("BlobConnectionString");

private static readonly string CosmosDbEndpoint = Environment.GetEnvironmentVariable("CosmosDbEndpoint");

private static readonly string CosmosDbKey = Environment.GetEnvironmentVariable("CosmosDbKey");

private static readonly string CosmosDbDatabaseId = Environment.GetEnvironmentVariable("CosmosDbDatabaseId");

private static readonly string CosmosDbContainerId = Environment.GetEnvironmentVariable("CosmosDbContainerId");

private static readonly string FormRecognizerEndpoint = Environment.GetEnvironmentVariable("FormRecognizerEndpoint");

private static readonly string FormRecognizerKey = Environment.GetEnvironmentVariable("FormRecognizerKey");

private static BlobServiceClient blobServiceClient = new BlobServiceClient(BlobConnectionString);

private static CosmosClient cosmosClient = new CosmosClient(CosmosDbEndpoint, CosmosDbKey);

private static DocumentAnalysisClient formRecognizerClient = new DocumentAnalysisClient(new Uri(FormRecognizerEndpoint), new AzureKeyCredential(FormRecognizerKey));

[FunctionName("QueueTriggerFunction")]

public static async Task Run([ServiceBusTrigger("image-processing-queue", Connection = "ServiceBusConnection")]Message message, ILogger log)

{

try

{

// Extract message content

string messageBody = System.Text.Encoding.UTF8.GetString(message.Body);

dynamic data = Newtonsoft.Json.JsonConvert.DeserializeObject(messageBody);

string imageName = data.imageName;

string language = data.language;

// Fetch the image from Blob Storage

BlobContainerClient containerClient = blobServiceClient.GetBlobContainerClient("incoming-images");

BlobClient blobClient = containerClient.GetBlobClient(imageName);

var blobStream = await blobClient.OpenReadAsync();

// Perform OCR using Form Recognizer

AnalyzeDocumentOperation operation = await formRecognizerClient.StartAnalyzeDocumentAsync("prebuilt-read", blobStream);

await operation.WaitForCompletionAsync();

AnalyzeResult result = operation.Value;

// Extract text and metadata

string extractedText = string.Join(" ", result.Pages.SelectMany(page => page.Lines).Select(line => line.Content));

var metadata = new { imageName, language, extractedText, timestamp = DateTime.UtcNow };

// Store the results in Cosmos DB

Container container = cosmosClient.GetContainer(CosmosDbDatabaseId, CosmosDbContainerId);

await container.CreateItemAsync(metadata, new PartitionKey(imageName));

// Move the image to the processed container

BlobContainerClient processedContainerClient = blobServiceClient.GetBlobContainerClient("processed-images");

BlobClient processedBlobClient = processedContainerClient.GetBlobClient(imageName);

await processedBlobClient.StartCopyFromUriAsync(blobClient.Uri);

await blobClient.DeleteAsync();

log.LogInformation($"Processed image: {imageName}");

}

catch (Exception ex)

{

log.LogError($"Error processing image: {ex.Message}");

throw; // This will trigger a retry and eventually move to the dead-letter queue if configured

}

}

}

```

### Environment Variables

- `BlobConnectionString`: Connection string for Azure Blob Storage.

- `ServiceBusConnection`: Connection string for Azure Service Bus.

- `CosmosDbEndpoint`: Endpoint for Azure Cosmos DB.

- `CosmosDbKey`: Primary key for Azure Cosmos DB.

- `CosmosDbDatabaseId`: Database ID in Cosmos DB.

- `CosmosDbContainerId`: Container ID in Cosmos DB.

- `FormRecognizerEndpoint`: Endpoint for Azure Form Recognizer.

- `FormRecognizerKey`: API key for Azure Form Recognizer.

### Monitoring and Maintenance

- \*\*Azure Monitor\*\*: Use Azure Monitor to track performance, detect issues, and set up alerts.

- \*\*Application Insights\*\*: Integrate Application Insights for detailed logging and diagnostics of Azure Functions.

- \*\*Cosmos DB Metrics\*\*: Monitor Cosmos DB metrics like RU/s consumption, latency, and storage usage to optimize performance and cost.

By following this architecture, you can build a robust and scalable solution for processing images with handwritten text using Azure services and Cosmos DB.

Certainly! I’ll create an architecture diagram using Azure Icons based on the provided input. Let’s break it down step by step:

1. **Blob Storage (incoming-images):** Represents the storage for incoming images.
2. **Service Bus (Queue):** Acts as a message queue.
3. **Azure Function (Queue Trigger):** Listens for messages from the queue.
4. **image-processing-queue:** Connects to both the Service Bus and the Azure Function.
5. **Message contains Image Name & Language:** Receives messages from the image-processing-queue.
6. **Fetch Image from Blob (incoming-images):** Retrieves images from Blob Storage.
7. **Perform OCR using Azure Cognitive Services:** Processes images for text extraction.
8. **Azure Cosmos DB:** Stores extracted text and metadata.
9. **Move Image to processed-images:** Stores processed images.
10. **Delete Image from incoming-images:** Removes images from the incoming storage.

Here’s the architecture diagram:

+--------------------+ +--------------------+ +--------------------+

| Blob Storage | | Service Bus | | Azure Function |

| (incoming-images) | | (Queue) | | (Queue Trigger) |

+--------------------+ +--------------------+ +--------------------+

| | |

| v v

| +-----------------------+ +------------------------+

| | image-processing | | Fetch Image from Blob |

| | -queue- | | (incoming-images) |

| +-----------------------+ +------------------------+

| | |

| v v

| +-----------------------+ +------------------------+

| | Message contains | | Perform OCR using |

| | Image Name & | | Azure Cognitive |

| | Language | | Services |

| +-----------------------+ +------------------------+

| | |

v v v

+--------------------+ +-----------------------+ +------------------------+

| Blob Storage | | Azure Cosmos DB | | Move Image to |

| (processed-images) | | | | processed-images |

+--------------------+ +-----------------------+ +------------------------+

| |

v v

+--------------------+ +-----------------------+

| Store Extracted | | Delete Image from |

| Text and Metadata | | incoming-images |

+--------------------+ +-----------------------+

Feel free to ask if you need any further details or adjustments! 😊