To set up batch transcription using Azure Whisper AI Speech for a call center with concurrency and specific customer handling, follow these steps:

**1. Set Up Azure Services**

1. **Create Azure Speech Resource:**
   * In the [Azure portal](https://portal.azure.com), create a Speech service resource if you haven't already.
   * Obtain the API Key and Endpoint URL.
2. **Create Blob Storage:**
   * Set up an Azure Storage account and a blob container to store the audio files.
   * Obtain the connection string for accessing Blob Storage.

**2. Configure the .NET Application**

1. **Install Necessary NuGet Packages:**
   * Install the Azure SDK packages for Blob Storage and REST API calls:

shell

Copy code

Install-Package Azure.Storage.Blobs

Install-Package Newtonsoft.Json

1. **Upload Audio to Blob Storage:**
   * Capture and upload live audio files to Azure Blob Storage. Here’s how to upload a file:

csharp

Copy code

using Azure.Storage.Blobs;

using System.IO;

using System.Threading.Tasks;

public async Task UploadAudioToBlob(string filePath, string blobName, string connectionString)

{

var blobServiceClient = new BlobServiceClient(connectionString);

var blobContainerClient = blobServiceClient.GetBlobContainerClient("audio-files");

await blobContainerClient.CreateIfNotExistsAsync();

var blobClient = blobContainerClient.GetBlobClient(blobName);

await blobClient.UploadAsync(filePath, true);

}

1. **Batch Transcription Using Whisper AI:**

**Prepare the Batch Transcription Request:**

* + Whisper API supports batch transcription via a REST API call. Submit a transcription job for each audio file.

**Submit Transcription Job:**

csharp

Copy code

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json.Linq;

public async Task<string> SubmitBatchTranscriptionJobAsync(string[] audioUrls, string subscriptionKey, string region)

{

var uri = $"https://{region}.api.cognitive.microsofttranslator.com/speech/recognition/batch/cognitiveservices/v1";

var body = new JObject

{

["audioUrls"] = new JArray(audioUrls),

["language"] = "en-US" // Adjust language as needed

};

using (var client = new HttpClient())

{

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);

var content = new StringContent(body.ToString(), Encoding.UTF8, "application/json");

var response = await client.PostAsync(uri, content);

var responseString = await response.Content.ReadAsStringAsync();

var jsonResponse = JObject.Parse(responseString);

return jsonResponse["jobId"].ToString(); // Retrieve the job ID for polling

}

}

**Poll for Job Status:**

csharp

Copy code

public async Task<JObject> CheckTranscriptionJobStatusAsync(string jobId, string subscriptionKey, string region)

{

var uri = $"https://{region}.api.cognitive.microsofttranslator.com/speech/recognition/batch/cognitiveservices/v1/{jobId}";

using (var client = new HttpClient())

{

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);

var response = await client.GetStringAsync(uri);

return JObject.Parse(response);

}

}

**Retrieve Transcription Results:**

csharp

Copy code

public async Task<JObject> GetTranscriptionResultsAsync(string jobId, string subscriptionKey, string region)

{

var uri = $"https://{region}.api.cognitive.microsofttranslator.com/speech/recognition/batch/cognitiveservices/v1/results/{jobId}";

using (var client = new HttpClient())

{

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);

var response = await client.GetStringAsync(uri);

return JObject.Parse(response);

}

}

1. **Store Transcription Results in Local Database:**
   * Save the transcription results into your local database. Here’s a basic example using Entity Framework:

csharp

Copy code

using System;

using System.Threading.Tasks;

using YourNamespace.Data; // Replace with your actual namespace

using YourNamespace.Models; // Replace with your actual namespace

public async Task SaveTranscriptionsAsync(string customerId, JObject transcriptions)

{

using (var context = new YourDbContext())

{

foreach (var transcription in transcriptions["results"])

{

var transcript = new Transcript

{

CustomerId = customerId,

Text = transcription["text"].ToString(),

Timestamp = DateTime.UtcNow

};

context.Transcripts.Add(transcript);

}

await context.SaveChangesAsync();

}

}

**3. Handle Concurrency**

1. **Batch Processing:**
   * For multiple audio files, ensure you batch process them efficiently. Use Azure Queue Storage or Azure Service Bus to queue and manage transcription requests.
2. **Asynchronous Operations:**
   * Use asynchronous methods for handling audio uploads, API requests, and database operations to support concurrent processing.

**4. Error Handling and Logging**

* **Error Handling:**
  + Implement error handling to manage failures in uploading, transcribing, or storing data. Log errors for troubleshooting.
* **Logging:**
  + Use a logging framework like Serilog or NLog to capture important events and errors.

**Example Main Method**

csharp

Copy code

using System;

using System.Threading.Tasks;

class Program

{

private static async Task Main(string[] args)

{

// Configuration (replace with actual values)

string blobConnectionString = "YourBlobConnectionString";

string speechSubscriptionKey = "YourSpeechSubscriptionKey";

string speechRegion = "YourSpeechRegion";

string[] audioFilePaths = { "path-to-audio1.wav", "path-to-audio2.wav" };

string blobContainerName = "audio-files";

string customerId = "customer-id";

// Upload audio files to Blob Storage

foreach (var filePath in audioFilePaths)

{

string blobName = System.IO.Path.GetFileName(filePath);

await UploadAudioToBlob(filePath, blobName, blobConnectionString);

}

// Generate URLs for audio files

var audioUrls = Array.ConvertAll(audioFilePaths, filePath =>

$"https://{blobConnectionString.Split(';')[0].Split('=')[1]}.blob.core.windows.net/{blobContainerName}/{System.IO.Path.GetFileName(filePath)}");

// Submit batch transcription job

string jobId = await SubmitBatchTranscriptionJobAsync(audioUrls, speechSubscriptionKey, speechRegion);

// Poll for job status and get results

JObject status;

do

{

await Task.Delay(5000); // Poll every 5 seconds

status = await CheckTranscriptionJobStatusAsync(jobId, speechSubscriptionKey, speechRegion);

} while (status["status"].ToString() != "Succeeded");

JObject transcriptionResults = await GetTranscriptionResultsAsync(jobId, speechSubscriptionKey, speechRegion);

// Save transcription results to local database

await SaveTranscriptionsAsync(customerId, transcriptionResults);

Console.WriteLine("Batch transcription process completed successfully.");

}

// Methods: UploadAudioToBlob, SubmitBatchTranscriptionJobAsync, CheckTranscriptionJobStatusAsync, GetTranscriptionResultsAsync, SaveTranscriptionsAsync

}

This example demonstrates the end-to-end process of batch transcription using Azure Whisper AI Speech. Adapt this template according to your specific requirements, such as handling large volumes of audio files, managing authentication securely, and integrating with your call center systems.