Armand Szilczer

DE6ZVJ

Contents

[Introduction 1](#_Toc163206726)

[Technologies Used 2](#_Toc163206727)

[Setup and Configuration 2](#_Toc163206728)

[Azure Blob/File Storage or Cosmos DB 3](#_Toc163206729)

[Creating a Container in Blob Storage: 3](#_Toc163206730)

[Application Architecture 4](#_Toc163206731)

[Implementation Details 5](#_Toc163206732)

[Handling Image Analysis 6](#_Toc163206733)

[Storing Data 7](#_Toc163206734)

[Testing and Results 7](#_Toc163206735)

[Interpretation of Results 10](#_Toc163206736)

[The original image: 10](#_Toc163206737)

[Conclusion 11](#_Toc163206738)

**Azure computer vision with blob file system**

# Introduction

Purpose of the Application

This application serves as a practical example of integrating cloud-based AI and storage services to process and manage data effectively. Specifically, it's designed to analyze images and store the results. The primary functionality is taking an image as input, using Azure Computer Vision to extract detailed information from the image, along with the certainty that the image is really what it thinks it is, and then storing this information in a cloud storage solution which is Azure Blob. *(Also I would like to state in the beginning that, while doing this application I was not concerned with security, as this is just school work and it’s only for demonstration and not a real-life application. In the future I and anyone who implements this should take security into account and make the application as secure as possible.)*

The use case for such an application is broad and applicable in various sectors. For instance, in the retail industry, it can be used to analyze product images for inventory management, in the security sector for surveillance purposes, or in social media platforms for content moderation and tagging. The application can also be employed in academic research for data collection and analysis, especially in fields where visual data is paramount.

# Technologies Used

*Azure Computer Vision:* This is a cloud-based service provided by Microsoft as part of its Azure suite. It offers pre-trained AI models that can analyze images and videos for content and other useful information. In our application, Azure Computer Vision is used to extract rich information from images such as tags, descriptions, and metadata. This service simplifies the process of implementing complex AI models for image analysis.

*Azure Blob/File Storage:* Azure Blob Storage is a scalable cloud storage service suitable for storing large amounts of unstructured data. In this application, it's used to store the results of the image analysis securely and efficiently. Blob Storage is ideal for this purpose due to its high availability, large capacity, and data redundancy features.

This application showcases the integration of advanced image processing with robust cloud storage solutions, demonstrating how modern cloud technologies can be utilized to build efficient and powerful data processing pipelines.

# Setup and Configuration

**Azure Computer Vision**

**Creating an Azure Account:**

If you don't already have an Azure account, start by signing up at the Azure Portal. You may be eligible for a free trial, which provides limited free access to Azure services.

Setting Up a Computer Vision Resource:

* Once logged in, navigate to the Azure Portal Dashboard.
* Click on “Create a resource” and search for “Computer Vision” in the Azure Marketplace.
* Select “Computer Vision” from the search results and click “Create”.
* Fill in the details such as the name of the resource, your preferred region, subscription plan, and resource group (you can create a new one if necessary).
* Review and confirm the selections and then click "Create". Azure will then deploy the Computer Vision resource.

Obtaining the Key and Endpoint:

* After the resource is deployed, go to the resource page in the Azure Portal.
* In the resource management pane, find the section named “Keys and Endpoint”.
* Here you will find two keys and an endpoint URL. Copy one of the keys and the endpoint URL, as they will be used in your application to authenticate and interact with the Computer Vision service.

# Azure Blob/File Storage or Cosmos DB

Setting Up Azure Blob/File Storage:

* Go to the Azure Portal Dashboard.
* Click on “Create a resource” and search for “Storage Account”.
* Select it and then click “Create”.
* Fill in the required details like account name, performance model, account kind, replication type, and location.
* Create or select an existing resource group, review your selections, and then click "Create".
* Once the storage account is created, navigate to it and find the “Access keys” section under “Security + networking”. Copy the connection string provided here.

# Creating a Container in Blob Storage:

* In the storage account, find the “Blob service” section and select “Containers”.
* Click “+ Container” to add a new one. Name your container (lowercase only) and set the public access level as per your requirement.
* Once the container is created, it’s ready to store blobs.

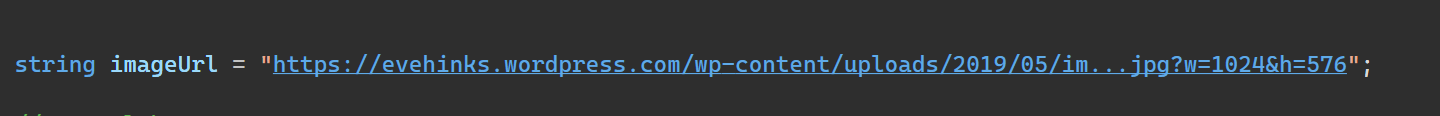
This setup will allow your application to interact with Azure's AI capabilities for image processing and provide scalable, secure storage solutions for handling the data generated.

# Application Architecture

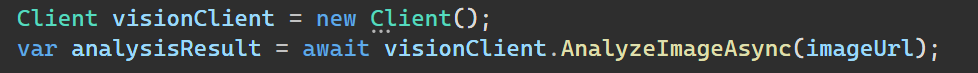
**Overview**

The application is designed as a cloud-based solution for processing and storing image data. Its architecture is centered around leveraging Azure's AI and storage services to analyze images and manage the extracted information.

**Description of Components**

*Image Input:* This is the initial step where an image is provided as input to the system. The source of the image can vary, such as a URL link, a file upload, or a direct capture from a camera. 

*Azure Computer Vision for Image Analysis:* Once the image is inputted, it's sent to Azure Computer Vision. This cloud-based AI service analyzes the image using advanced algorithms and machine learning models. It can extract a range of information such as tags, descriptions, and objects within the image.



*Data Extraction:* After the image is processed, Azure Computer Vision outputs detailed data about the image. This data typically includes a JSON structure with tags and descriptions, metadata like dimensions and format, and any other relevant analysis results.

*Azure Blob:*  If Blob Storage is chosen, the extracted information is serialized into a JSON file and stored in a specified container within the Blob Storage. This service is well-suited for unstructured data and is scalable, secure, and highly available.

A screenshot of a computer program

Description automatically generated

This architecture allows the application to efficiently process and store large volumes of image data while leveraging the powerful cloud services provided by Azure.

# Implementation Details

Code Explanation

The provided code is structured into several key sections, each handling different aspects of the application's functionality:

Client Class:

The Client class is responsible for interfacing with Azure's Computer Vision API.

It initializes a ComputerVisionClient object with the necessary credentials and endpoint.

The AnalyzeImageAsync method is the core function, taking an image URL as input and returning the analysis results. It configures the types of visual features to be analyzed (like tags and descriptions) and calls the Azure Computer Vision API. A screen shot of a computer program

Description automatically generated

Blob Class:

The Blob class deals with storing data in Azure Blob Storage.

UploadToBlobAsync is the main method, taking the connection string, container name, blob name, and the image analysis results as input.

This method serializes the ImageAnalysis object to JSON, converts it to a byte array, and uploads it to a specified blob within a container in Azure Blob Storage. A black background with many lines

Description automatically generated with medium confidence

# Handling Image Analysis

The image analysis is handled by the Azure Computer Vision service.

The process begins with the Client class's AnalyzeImageAsync method, which initializes the Computer Vision client using the provided API key and endpoint.

The image URL is passed to AnalyzeImageAsync, which requests Azure Computer Vision to analyze the image for specified visual features.

Once the analysis is complete, Azure returns an ImageAnalysis object containing various pieces of information extracted from the image, such as tags, descriptions, and other metadata.

A screen shot of a computer code

Description automatically generated

# Storing Data

Azure Blob/File Storage:

After receiving the analysis results, the application uses the Blob class to store this data.

The UploadToBlobAsync method in the Blob class takes the analysis results, serializes them into JSON format, and stores them in Azure Blob Storage.

The method creates a new Blob Service Client with the provided connection string, checks for the specified container (or creates it if it doesn’t exist), and then creates or overwrites the blob with the name specified.

The JSON data is converted into a byte array and then uploaded to the blob.

# Testing and Results

Testing Procedure

To thoroughly test the application, the following steps were undertaken:

Selecting Test Images:

A few images were tested, mostly animals, the result that I will be showcasing is the analysis of a hedgehog.

Running the Application:

Each image was inputted into the application through its URL.

The application then processed these images using the Azure Computer Vision API.

Verifying Storage:

After processing each image, the results were automatically stored in Azure Blob Storage.

The storage was then checked to ensure that the data was correctly serialized and uploaded.

Edge Cases:

The application was also tested with some edge cases, such as invalid image URLs and images that are difficult to analyze (like those with low resolution or ambiguous content), to observe how the system behaves under these conditions.

Results

One of the images used for testing was a picture of a hedgehog. The Azure Computer Vision API analyzed this image and produced the following results, which were then stored in Azure Blob Storage:

{

    "Categories": null,

    "Adult": null,

    "Color": null,

    "ImageType": null,

    "Tags": [

        {

            "Name": "animal",

            "Confidence": 0.9999550580978394,

            "Hint": null

        },

        {

            "Name": "mammal",

            "Confidence": 0.999904215335846,

            "Hint": null

        },

        {

            "Name": "hedgehog",

            "Confidence": 0.9974547624588013,

            "Hint": null

        },

        {

            "Name": "erinaceidae",

            "Confidence": 0.9600135087966919,

            "Hint": null

        },

        {

            "Name": "porcupine",

            "Confidence": 0.9448654651641846,

            "Hint": null

        },

        {

            "Name": "outdoor",

            "Confidence": 0.8603819012641907,

            "Hint": null

        },

        {

            "Name": "grass",

            "Confidence": 0.7202334403991699,

            "Hint": null

        },

        {

            "Name": "wildlife",

            "Confidence": 0.6989262104034424,

            "Hint": null

        },

        {

            "Name": "ground",

            "Confidence": 0.6047341823577881,

            "Hint": null

        },

        {

            "Name": "bird",

            "Confidence": 0.5449870824813843,

            "Hint": null

        }

    ],

    "Description": {

        "Tags": [

            "grass",

            "hedgehog",

            "outdoor",

            "little"

        ],

        "Captions": [

            {

                "Text": "a hedgehog on a white surface",

                "Confidence": 0.4997209310531616

            }

        ]

    },

    "Faces": null,

    "Objects": null,

    "Brands": null,

    "RequestId": "ed625aa8-27e7-445e-88c5-71b76b09bdc1",

    "Metadata": {

        "Width": 1024,

        "Height": 576,

        "Format": "Jpeg"

    },

    "ModelVersion": "2021-05-01"

}

# Interpretation of Results

*Tags:* The tags section contains keywords identified in the image, each accompanied by a confidence score. High confidence scores, like those for "animal", "mammal", and "hedgehog", indicate the model's high certainty about these elements in the image.

*Description:* The description section provides a textual summary of the image. In this case, "a hedgehog on a white surface" with a moderate confidence score indicates a recognized scene but with some uncertainty, likely due to factors in the image's composition.

*Metadata:* This includes basic information about the image such as its width, height, and format.

*ModelVersion:*  Indicates the version of the Computer Vision model used for analysis.

These results demonstrate the capability of Azure Computer Vision to accurately identify and describe elements in an image. They also show the effectiveness of the application in processing and storing these insights, making them available for various applications, such as content categorization, automated tagging, and more.

The original image: A hedgehog on a table

Description automatically generated

# Conclusion

The application effectively demonstrates how the integration of cloud-based AI and storage devices is easy and how they can manage image data. By utilizing Azure’s Computer Vision, the application can analyze images of any kind and give a detailed overview of what is on the image with a given certainty, in this case, stored as a JSON file in an Azure Blob file storage, however, it can be used with different approaches.

The application stands as a testament to how useful and comprehensive these cloud-based AI solutions are, they are efficient, easy-to-use, scalable solutions.

The application can further be improved by adding some or all of the following features:

* Extend the analysis features
* User interface
* Real-time Processing
* Data visualization
* Integration with other Azure services
* Privacy and security enhancement
* Scalability and performance optimizations

*(Also I would like to state in the beginning that, while doing this application I was not concerned with security, as this is just school work and it’s only for demonstration and not a real-life application. In the future I and anyone who implements this should take security into account and make the application as secure as possible.)*