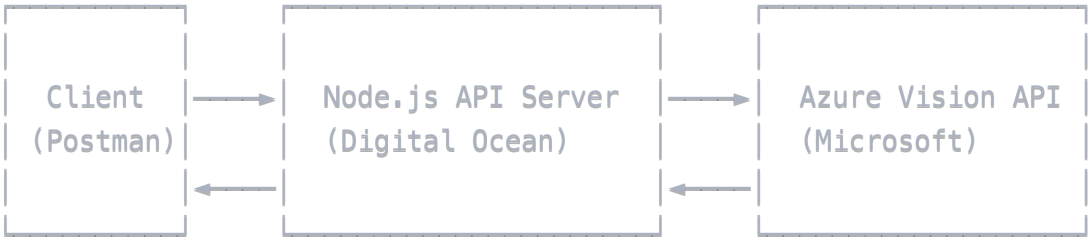
API Documentation: Image Analysis with Azure Vision API

Overview

This documentation provides comprehensive information about the image analysis API that serves as an intermediary between users and the Azure Vision API. The API allows users to upload images and receive detailed analysis results without having to register with or configure Azure services directly.

Technical Architecture

The API is built using Node.js and Express, running on a CentOS server on Digital Ocean. It handles image uploads, communicates with the Azure Vision API, and returns the analysis results to the client.



Server Configuration

- **Platform**: Digital Ocean Droplet
- **Operating System**: CentOS
- **Process Manager**: PM2
- **Server**: Node.js with Express
- **IP Address**: 161.35.135.31
- **Port**: 3000

API Endpoints

1. Analysis Endpoint

URL: `http://161.35.135.31:3000/analyze`

Method: POST

Content-Type: multipart/form-data

Request Parameters:

- `image` (required): The image file to be analyzed

Example Request using Postman:

1. Set request type to POST
2. Enter URL: `http://161.35.135.31:3000/analyze`
3. In the Body tab, select "form-data"
4. Add a key named "image" and set its type to "File"
5. Upload your image file
6. Click "Send"

Example Postman Configuration:

```
POST http://161.35.135.31:3000/analyze
Body:
  - form-data
    - key: image (type: File)
    - value: [your uploaded image]
```

Example Response:

json


```
{
    "tags": [
        {
            "name": "text",
            "confidence": 0.996180534362793
        },
        {
            "name": "screenshot",
            "confidence": 0.9577434062957764
        },
        {
            "name": "design",
            "confidence": 0.422126442193985
        }
    ],
    "description": {
        "tags": [
            "timeline"
        ],
        "captions": [
            {
                "text": "timeline",
                "confidence": 0.34569695591926575
            }
        ]
    },
    "requestId": "8bad07f9-3131-47d4-9de4-43dea916b084",
    "metadata": {
        "height": 1036,
        "width": 1592,
        "format": "Png"
    },
    "modelVersion": "2021-05-01"
}
```


2. Health Check Endpoint

URL: `http://161.35.135.31:3000/`

Method: GET

Response: "Azure Vision API service is running"

How the API Works

1. **Image Upload**: When a client uploads an image using the `/analyze` endpoint, the server receives the image through a multipart form-data request.
2. **Image Processing**: The uploaded image is temporarily saved on the server.
3. **Azure Communication**: The server sends the image to the Azure Vision API using the following endpoint:

   ```
   https://ai-rmannin42248ai836358025982.cognitiveservices.azure.com/vision/v3.2/analyze?visualFeatures=Description,Tags
   ```

4. **Analysis Results**: Azure Vision API analyzes the image and returns the results, which include:
   - Tags identifying the contents of the image
   - A descriptive caption for the image
   - Confidence scores for each tag and caption
   - Metadata about the image (dimensions, format)
5. **Response to Client**: The analysis results are forwarded to the client.
6. **Cleanup**: The temporarily saved image is deleted from the server.

Azure Vision API Integration

This API integrates with Azure's Computer Vision service, specifically the Vision v3.2 API. The integration uses the following components:

1. **Authentication**: Uses an Azure subscription key for authentication.
2. **Endpoint**: Connects to the East US 2 region endpoint.
3. **Features**: Utilizes the Description and Tags features of the Azure Vision API.

Implementation Details

Server Implementation

The server is implemented using Node.js and Express. Key libraries used include:

- **express**: Web framework for handling HTTP requests
- **multer**: Middleware for handling multipart/form-data
- **axios**: HTTP client for making requests to Azure
- **dotenv**: For managing environment variables

- **fs**: File system module for reading and writing files

Code Structure

```javascript
// Load environment variables
require('dotenv').config();

// Import required modules
const express = require('express');
const multer = require('multer');
const fs = require('fs');
const axios = require('axios');

// Initialize Express application
const app = express();
const upload = multer({ dest: 'uploads/' });
const port = process.env.PORT || 3000;

// Ensure uploads directory exists
if (!fs.existsSync('uploads')) {
  fs.mkdirSync('uploads');
}

// Define analyze endpoint
app.post('/analyze', upload.single('image'), async (req, res) => {
  // Process image upload
  // Send to Azure Vision API
  // Return results to client
});

// Define health check endpoint
app.get('/', (req, res) => {
  res.send('Azure Vision API service is running');
});

// Start the server
app.listen(port, '0.0.0.0', () => {
  console.log(`Server running on http://0.0.0.0:${port}`);
});
```

Deployment

The API is deployed on a Digital Ocean droplet and managed using PM2 to ensure the service remains running. The deployment process includes:

1. Setting up a CentOS server on Digital Ocean
2. Installing Node.js and necessary dependencies
3. Creating the application files
4. Configuring environment variables
5. Starting the application with PM2

Error Handling

The API includes robust error handling for various scenarios:

1. **Missing Image**: If no image is uploaded, the API returns a 400 error with a message.
2. **Azure API Errors**: If there's an issue with the Azure Vision API, the error details are logged on the server, and a 500 error is returned to the client.
3. **Network Issues**: The API handles connection problems with appropriate error messages.

Security Considerations

1. **API Key Protection**: The Azure subscription key is stored in environment variables, not in the code.
2. **Temporary File Handling**: Uploaded images are deleted after processing to avoid storage issues and potential security risks.
3. **Error Message Sanitization**: Error messages are sanitized before being sent to clients to prevent information leakage.

Testing

The API can be tested using tools like Postman:

1. Create a new request in Postman
2. Set the method to POST
3. Enter the URL: `http://161.35.135.31:3000/analyze`
4. In the Body tab, select "form-data"
5. Add a key named "image" and set the type to File
6. Upload an image file
7. Click Send to receive the analysis results

Troubleshooting

Common issues and their solutions:

1. **500 Internal Server Error**: Check server logs for details about the error.
2. **401 Unauthorized**: Verify that the Azure subscription key is correct.
3. **404 Not Found**: Ensure the correct API endpoint is being used.

Conclusion

This API provides a simple yet powerful way to access Azure Vision API's image analysis capabilities without requiring direct configuration of Azure services. By handling the authentication and communication with Azure, it simplifies the process for end users who want to analyze images.