Azure Image Processor – Serverless Blob Trigger Function

Technologies Used: Azure Functions, Azure Blob Storage, Python, Visual Studio Code, Azure CLI, Git, GitHub

---

Project Overview:

Built a serverless image processing pipeline using Azure Functions and Blob Storage to automate the transformation of uploaded images. The application listens for new image uploads in a storage container, converts them to grayscale, and saves the processed output to a separate container using an event-driven function written in Python.

Key Features:

Developed a Blob Trigger Function in Python that activates automatically when an image is uploaded to the `input-images` container in Azure Blob Storage.

Converted uploaded images to grayscale using the Pillow (PIL) library and saved the output to a `processed-images` container.

Configured the project using `function.json` to define input/output bindings and used `host.json` for Azure Function runtime settings.

Used `requirements.txt` to manage dependencies and `.gitignore` to exclude local files from version control.

Secured the connection string using the `local.settings.json` file during local development (excluded from GitHub).

Built and tested the function locally using Visual Studio Code and deployed it to Azure using the Azure CLI.

Pushed the complete project structure to GitHub with a clean and modular directory layout for easy reuse and scalability.

Code Structure (ImageProcessorFunction/init.py):

Import Libraries

- `azure.functions` for Azure Function bindings
- `PIL.Image` and `io` for image processing and streaming

Define Main Function

- `main(myblob, outputBlob)` processes the uploaded image.
- Reads the image from the input stream.
- Converts it to grayscale.
- Writes the processed image to the output blob stream.

Trigger & Binding

- Trigger: Upload to `input-images/{name}`
- Output: Saves to `processed-images/{name}`