

**Table of Contents**

1. Introduction
2. Objective
3. Why I Built This Project
4. Technology Used
5. Project Folder Structure
6. Setup Checklist
7. Processing Flow
8. Step-by-Step Implementation
   * A. image-resizer
   * B. image-uploader
9. Output & Testing
10. Conclusion

# Introduction

This project is a serverless image processing application built using AWS Lambda and S3. It automatically resizes and optimizes images uploaded to an S3 bucket using Python and the Pillow library. The processed images are then stored in a separate destination S3 bucket.

# 2. Objective

To automate image optimization and resizing using a serverless architecture, eliminating the need for manual intervention or dedicated servers.

# 3. Why I Built This Project

* To gain hands-on experience with AWS Lambda and event-driven architecture.
* To automate repetitive image handling tasks.
* To apply Python and cloud knowledge to a real-world problem.

# 4. Technology Used

* + AWS Lambda
  + Amazon S3
  + Python 3.11
  + Pillow Library
  + HTML/CSS/JavaScript (for frontend)

# 5. Folder Structure

Project-1/  
├── image-resizer/  
│ ├── lambda\_function.py  
│ ├── requirements.txt  
│ └── image-resizer.zip  
│  
├── image-uploader-frontend/  
│ ├── index.html  
│ ├── script.js  
│ └── style.css

# 6. Setup Checklist

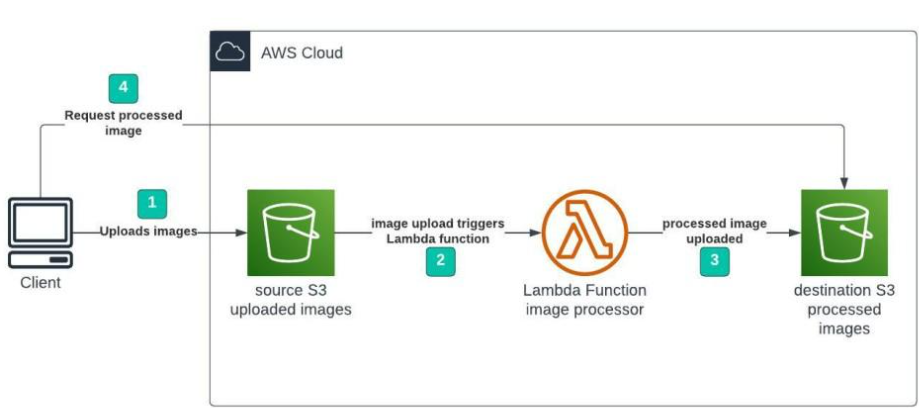
## A. Hardware

* Window 10 or above
* 4 GB RAM or higher

## B. Software

* + Python Installed
  + AWS CLI Configured
  + Notepad / VS Code

# Steps7. Processing Flow



# 8. Implementation Steps

## A. Image-resizer

**1. Create two S3 buckets**

* kashish-source-bucket
* Kashish-destination-bucket

**2. Create `lambda\_function.py` with the image resizing logic using Pillow.**



**3. Create `requirements.txt` with `Pillow`.**



**4. Install dependencies into the folder.**



**5. Create a deployment package using:**



**6. Upload to Lambda**

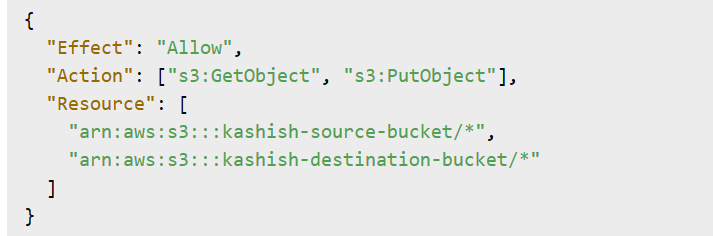
* Runtime: Python 3.11
* Handler: “lambda\_function.lambda\_handler”
* Set environment variable: “DEST\_BUCKET” = “kashish-destination-bucket”.

**7. Add S3 Trigger**

Source: “kashish-source-bucket”

Event: “PUT”

**8. IAM Role with permissions**:



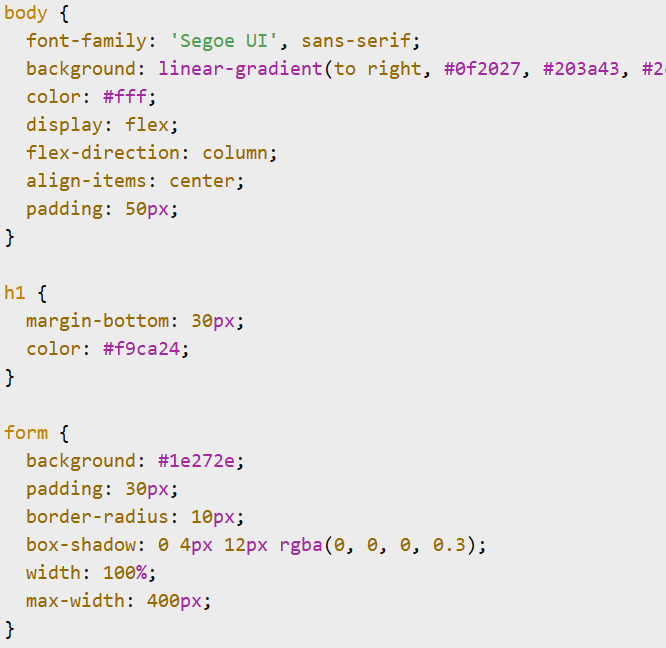
## B. Image-uploader-frontend

1. HTML form in `index.html` for image upload

A screenshot of a computer program

AI-generated content may be incorrect.

2. CSS in `style.css` for a modern and attractive interface.



The code is continued………

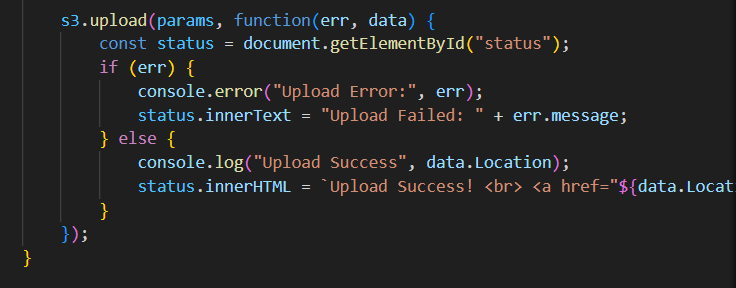


A white background with black text

AI-generated content may be incorrect.

3. JavaScript (`script.js`) for handling file selection and (future) upload.





# 9. Output & Testing

* + Upload any image (JPG/PNG) to `kashish-source-bucket`.
  + Lambda resizes and stores it in `kashish-destination-bucket` with name `resized-<filename>.jpg`.
  + Bucket policies updated to allow browser access.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a bucket

AI-generated content may be incorrect.

# 10. Conclusion

This project showcases the use of AWS Lambda for real-time, serverless image optimization. It is scalable, cost-effective, and integrates well with frontend interfaces for a seamless user experience.