CSL7090 SDE Mini Project

SERVERLESS IMAGE PROCESSING APPLICATION

Supervisior: Dr. Sumit Kalra

Team Members:
Ayush Jain (B21BB006)
Lokesh Kiran Chaudhari (B21CS041)
Shrashti Saraswat (B21CS081)

MAIN IDEA

The **Serverless Image Processing Application** leverages the power of cloud computing to perform image transformations (such as resizing, greyscale, and blur) in a scalable, efficient, and cost-effective manner.

By utilizing **AWS Lambda** and **S3**, this project offloads the complexities of resource management, enabling the application to handle varying workloads without manual intervention.

The application is designed to provide a seamless user experience by automatically scaling the infrastructure based on demand, making it highly efficient for unpredictable workloads.

KEY FEATURES

SEAMLESS IMAGE UPLOAD

Users can upload images through a web interface, which are then stored in Amazon S3.

MULTIPLE PROCESSING TASKS

- Resizing: The application can resize images based on user-defined dimensions.
- Filtering: Users can apply filters to images.

SERVERLESS ARCHITECTURE

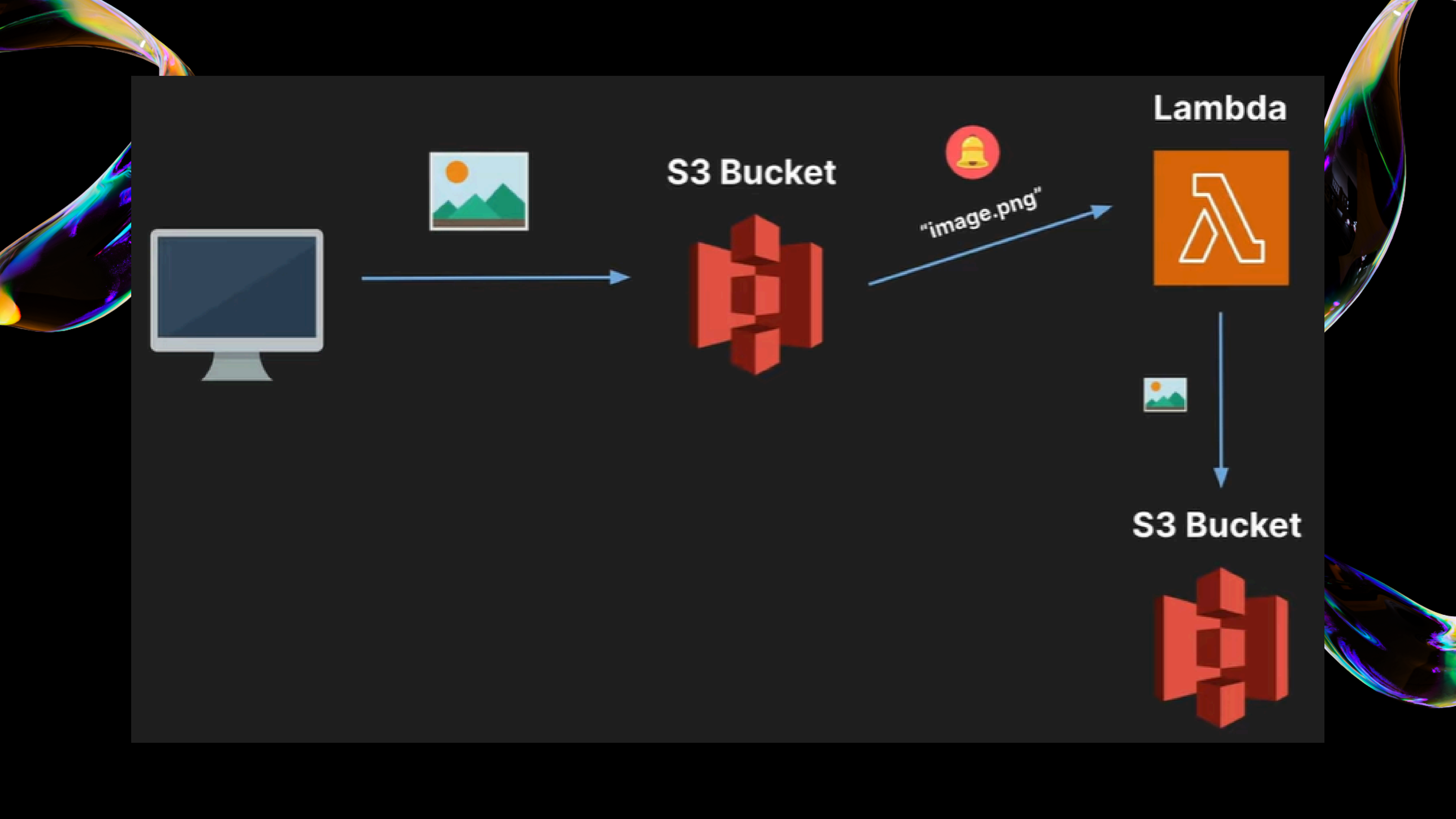
The application is built using AWS Lambda, ensuring that compute resources automatically scale based on the number of image processing requests.

EVENT-DRIVEN PROCESSING

Image processing tasks are triggered by events such as API requests or S3 uploads, ensuring a highly responsive and reactive system.

PROJECT ARCHITECTURE OVERVIEW

- **Client:** Users interact with a front-end to upload images and select processing tasks.
- **API Gateway:** Acts as the interface between the client and AWS Lambda.
- **AWS Lambda:** Handles image processing tasks (resize, greyscale, and blur) in a stateless, serverless manner.
- **Amazon S3:** Stores both original and processed images.
- **Pattern Used:** The project employs a combination of Client-Server and Monolithic architectural patterns.


Lambda
S3 Bucket
"image.png"
S3 Bucket

ARCHITECTURAL PATTERNS USED

Client-Server Pattern

Client:

Users interact with the front-end to upload images and select processing options.

Server:

AWS services such as API Gateway, Lambda, and S3 handle requests, image processing tasks and storage.

Advantages

Scalability: The server-side (AWS) can scale automatically based on the number of requests, making the application highly efficient and responsive to user demand.

Efficiency: With serverless architecture, the system scales dynamically without requiring manual provisioning.

Monolithic Pattern

Definition

All image processing logic (resizing, filtering) is bundled into a single Lambda function.

Advantages

Simplified Deployment: Having all logic in a single function makes deployment and development simpler.

Less Overhead: With a single function handling multiple tasks, it reduces the complexity of managing multiple services and routes.

COMPARISON BETWEEN MONOLITHIC AND CLIENT-SERVER PATTERNS

MONOLITHIC PATTERN

Pros:

- Simplicity: Easier to deploy, manage, and update the entire application as it resides in a single Lambda function.
- Low Initial Complexity: Suitable for smaller projects or when the application logic is not too complex.

Cons:

- Scaling Limitations: As the application grows and the number of requests increases, it becomes harder to scale individual components.
- Harder Maintenance Over Time: Adding new features or fixing bugs becomes more complex as all logic resides in one function.

CLIENT-SERVER PATTER

Pros:

- Modular Scalability: The clear separation of client and server allows each to scale independently. AWS Lambda can scale based on demand without affecting the front-end.
- Maintainability: Easier to maintain and expand the system. Processing tasks can be separated into distinct services in the future.

Cons:

- Initial Complexity: Involves more initial setup (API Gateway, Lambda functions, S3), which may be overkill for very simple applications.
- Higher Deployment Overhead: More components to manage compared to a monolithic design, especially if broken down into smaller services.

AWS DEPLOYMENT

Amazon S3

S3 is used to store both the original images uploaded by users and the processed images (resized, filtered, converted).

S3’s durability and scalability ensure that images are always available without the need for managing storage infrastructure.

AWS Lambda

Lambda functions handle image processing tasks (resize, greyscale, and blur) in a stateless, serverless manner.

Lambda scales automatically based on the number of requests, ensuring that the application can handle large workloads efficiently.

API Gateway

API Gateway provides the interface for users to upload images and trigger Lambda functions for processing. It securely exposes the Lambda functions to the web.

Event-Driven

The architecture is event-driven, where S3 or API Gateway events trigger the execution of Lambda functions, ensuring an optimized, real-time processing workflow.

CLOUD VS. NON-CLOUD SETUP

Cloud Setup (AWS Lambda + S3)

Automatic Scaling

AWS Lambda scales automatically in response to demand, handling varying workloads without manual intervention.

Cost Efficiency

With a pay-as-you-go model, you are only charged for the compute and storage resources used, avoiding upfront costs.

High Availability

AWS services are distributed across multiple availability zones, ensuring fault tolerance and uptime.

Non-Cloud Setup (On-Premise)

Manual Scaling

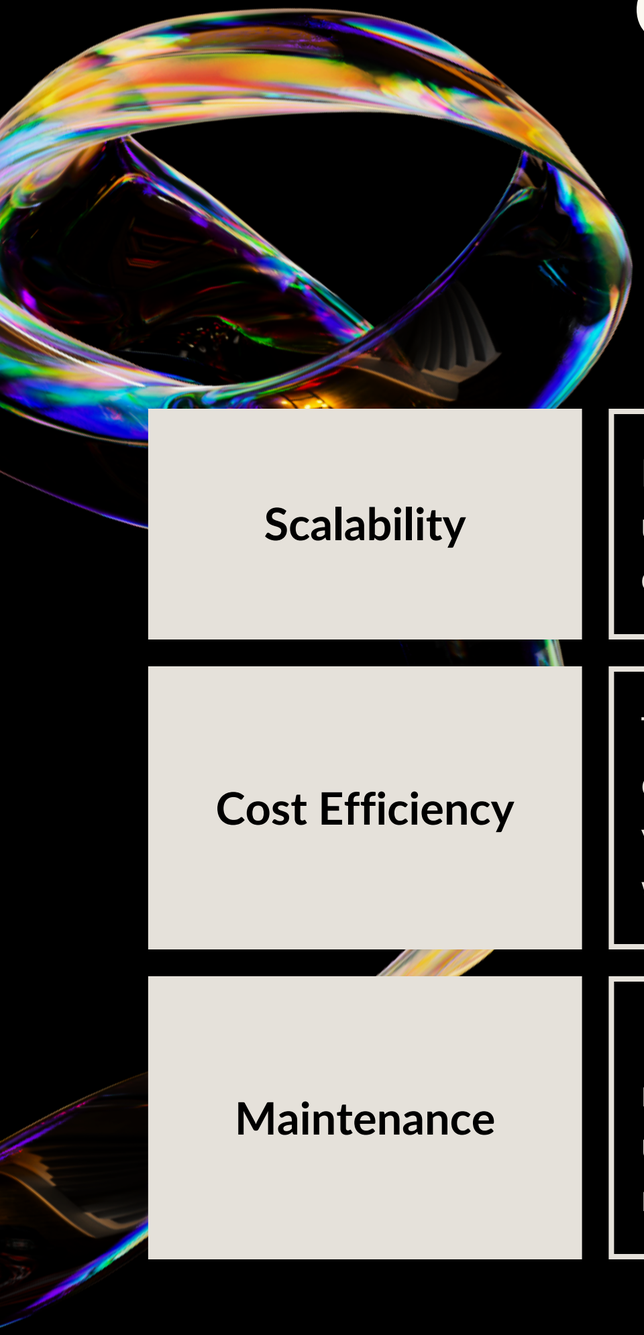
Scaling requires manually provisioning additional hardware or servers to handle increased workloads.

Fixed Costs

High upfront costs for purchasing and maintaining physical servers, regardless of how much they are used.

Limited Availability

On-premise setups require manual failover and backup solutions to handle system failures, leading to potential downtime.



COMPARISON BETWEEN CLOUD AND NON-CLOUD SETUP

	Cloud Setup	Non-Cloud Setup
Scalability	**Dynamically scales to handle increased demand using AWS Lambda. There is no need to manage or provision resources manually.**	**Scaling requires adding or provisioning more hardware manually, which is time-consuming and resource-intensive.**
Cost Efficiency	The pay-as-you-go model makes cloud more cost-efficient, as you only pay for the resources you use. It's especially beneficial for applications with variable workloads.	Involves high fixed costs for purchasing, maintaining, and powering servers. Even during idle times, these costs remain.
Maintenance	Minimal maintenance required since AWS manages the infrastructure. Tasks like software updates, security patches, and server management are handled by AWS.	Requires significant manual maintenance, including hardware management, software updates, and security patches. This increases operational complexity and costs.

THANK YOU