Assignment 3 Report

Viet Hoang Pham, Tran Anh Thu Pham, Phong Tran

COS20019 - Cloud Computing Architecture

Swinburne University of Technology

104506968@student.swin.edu.au ; 103818400@student.swin.edu.au; 104334842@student.swin.edu.au

Abstract— This report presents a serverless cloud architecture specifically designed for the Photo Album application. The architecture aims to transition to managed cloud services, ensure scalability to handle growing demand, make use of serverless computing, and replace the conventional relational database. The suggested design integrates multiple AWS services, such as S3, DynamoDB, Lambda, Route 53, SNS, SQS, Rekognition, MediaConvert, and API Gateway. The main goals are to maximize the efficiency of uploading photos and videos, automate the processing of material, and improve response times worldwide. The study provides a comprehensive analysis of the architecture's structure, essential use cases, and a summary of the integrated AWS services. Furthermore, it offers a deeper understanding of the reasoning behind the design, showcasing how it fulfills business needs and design criteria in order to give a strong, adaptable, and quick-to-respond solution for the Photo Album application.

Keywords— serverless, event-driven, S3 bucket, EC2 instance, thumbnails, reformatting, transcoding, Lambda, reprocessing media, decoupling, cloud service

I. Introduction

The exceptional success of the Photo Album program has sparked enthusiasm and presented us with fresh obstacles as we endeavour to satisfy the escalating requirements of our expanding user community. Given the current rate of user growth, which is projected to persist for the next two to three years, it is crucial to implement a strong and adaptable infrastructure. Important areas for improvement involve utilizing managed cloud services to reduce the need for in-house system administration, improving reaction times on a global scale, maximizing computing resources, and migrating to a serverless/event-driven design. In addition, it is imperative that we improve our media processing capabilities to effectively manage different forms and be prepared for future expansions, such as video media.

The primary objective of our design proposal is to leverage AWS services, specifically S3 for media storage, while also investigating cost-efficient alternatives to our existing relational database. By adopting a decoupled design for media reformatting and transcoding, we can guarantee that the application maintains its responsiveness and scalability. This strategy will entail initiating the automatic generation of alternative media versions when uploading, utilizing a versatile framework that enables future expansions like AI-powered tagging, and distributing processing jobs across appropriate platforms such as EC2 instances and AWS Lambda. This study offers our comprehensive design plan and provides a rationale for why it is the optimal choice for achieving the company's objectives and facilitating future expansion.

II. Architecture Design

A. Architecture diagram

This diagram below (Figure 1) illustrates the serverless cloud architecture for the Photo Album application, leveraging various AWS services to achieve a scalable, secure, and efficient solution. Our solution is divided into four tiers: security and monitoring, presentation, logic, and data.

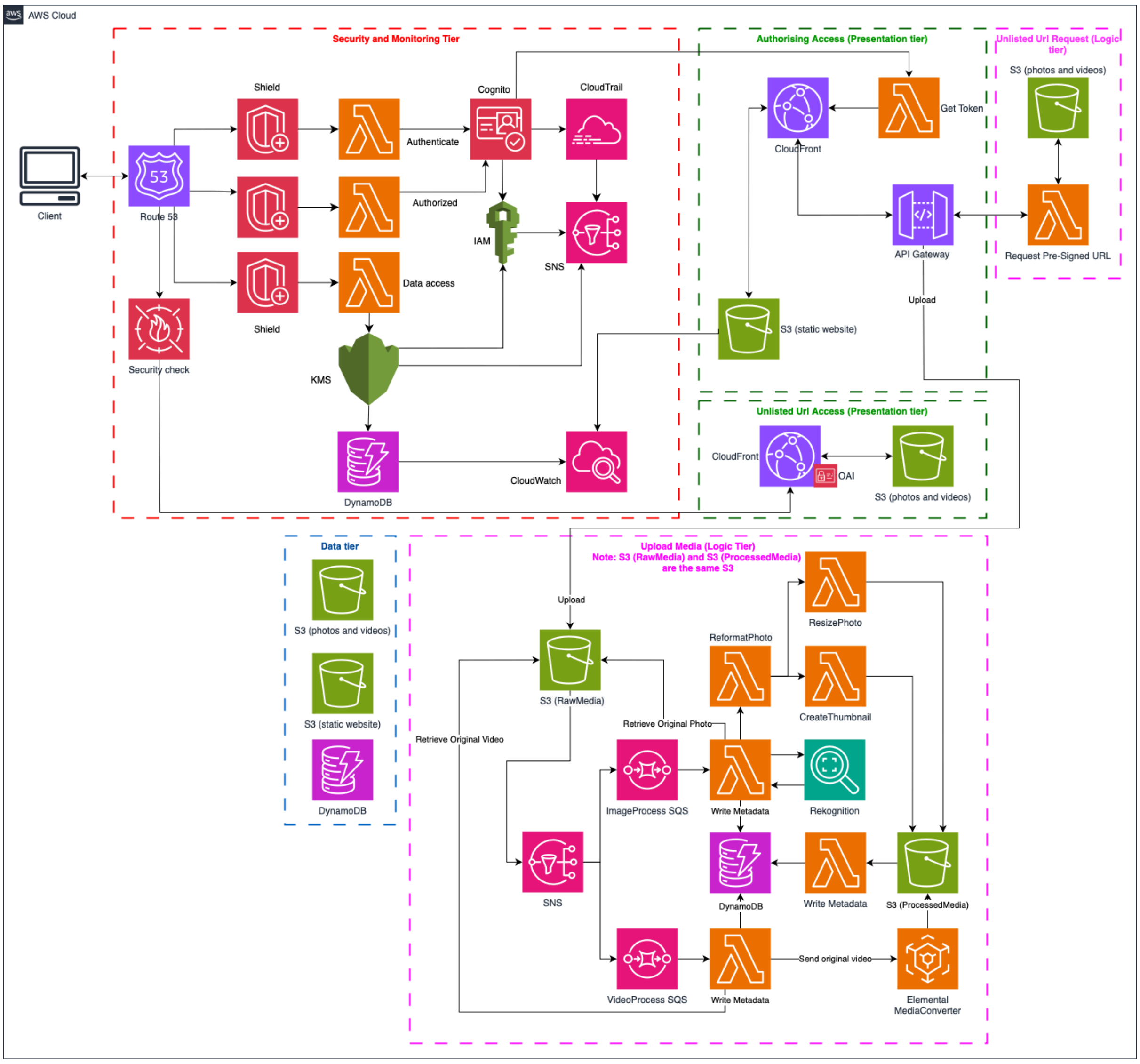


Fig. 1 AWS Photo Album architectural diagram

B. Key Services used in the Proposed Solution

1) Security and Monitoring

a) Authorization: IAM

Functionality and reason of using: Firstly, a user uses their credentials to authenticate with AWS by matching the credentials to a trusted principal. Then, a request for resource access will be made, which AWS responds to by checking if the principal is authorized. This involves verifying the identity against policies that control access levels. If authorized, the user can perform actions on AWS resources, such as launching EC2 instances or deleting S3 buckets.[1]

b) Authentication: AWS Cognito

Functionality: Amazon Cognito is an identity platform for web and mobile apps, serving as a user directory and authentication server. It supports authentication through its user directory, and identity providers such as Google and Facebook.

Reason of using: Firstly, the user signs in through a user portal and gets OAuth 2.0 tokens. Then, the app exchanges these tokens with an identity pool for credentials. These credentials are assigned to the user's session, enabling access to AWS services such as S3 or DynamoDB.[2]

c) Authentication: AWS Simple Notification Services (SNS)

Functionality: Amazon Simple Notification Services (SNS) is a managed service facilitating message delivery between publisher and subscriber. Publishers send messages to a topic, which acts as a logical access point and communication channel for asynchronous communication with subscribers.

Reason of using: Clients can subscribe to an SNS topic to receive messages through endpoints such as AWS SQS, AWS Lambda, email, etc.[5]

d) Protected stores: AWS KMS and AWS CloudTrail

Functionality: AWS Key Management Service (KMS) provides centralized control over cryptographic keys used to secure data. It integrates with other AWS services, especially the process of encrypting data stored in these services and managing access to the keys needed for decryption.[3]

Reason of using: AWS CloudTrail is integrated with AWS KMS, enabling administrators to audit key usage, including which keys were used, on which resource, and when.[3]

e) Enforcement: AWS Shield (optional due to high cost)

Functionality: AWS Shield Standard protects against Distributed Denial of Service (DDoS) attacks for AWS resources at the network layer and transport layer (layers 3 and 4) as well as the application layer (layer 7).

Reason of using: In a DDoS attack, numerous compromised systems generate excessive traffic to overwhelm the target, which can block users from accessing the services and cause the target to crash due to the heavy traffic load.[4]

f) Monitoring: AWS CloudWatch

Functionality and reason of using: AWS CloudWatch works as a metric repository. For example, EC2 instance sends metrics to this repository, allowing retrieval of statistics based on those metrics. Metrics can be used to calculate statistics and display data in the CloudWatch console. Alarms can be configured in CloudWatch to automatically stop, and start, and terminate an EC2 instance based on specific conditions. Alarms can trigger actions in AWS EC2 Auto Scaling and AWS SNS.[6]

2) *Presentation Tier (Accessing static website and photos with signed URL)*

a) *Route 53*

Functionality: Amazon Route 53 is a reliable and flexible Domain Name System (DNS) solution that handles heavy traffic, and connects between user requests and internet applications hosted on AWS or on-premises. It offers features and capabilities, such as domain registration, customising DNS routing policies, or distributing domain name queries on global traffic management. Route 53 ensures high-availability by resource health checks, supporting geolocation and latency-based routing. Therefore, it is essential for managing incoming web traffic across AWS services while being able to provide fault tolerance, which increases end-users' experience. It also supports DNSSEC for enhancing security and integrates seamlessly with other AWS services, making it vital for globally distributed websites and applications.[7]

Reason of using: When it comes to our architecture, route 53 becomes a vital component to ensure high availability and DNS resolution at a low-latency. Additionally, in terms of routing users to the nearest location, it becomes an important role to enhance the responsiveness of Photo Album web application, which increase the capability standard for adapting to user's demand, therefore, increasing overall user's experience.

b) *CloudFront*

Functionality: Amazon CloudFront is a CDN (content delivery network) that ensures high transfer speeds and low latency at delivering data distribution of static web content, like the files of .html and media assets, to users at a nearest edge locations, also called the global network of data centers. Key features include caching, edge compute capabilities, and WebSockets support. It also offers strong security measures like traffic encryption, access controls, increasing efficiency and better static content distribution, therefore, improving the end-user experience. [8], [9]

Reason of using: In order to optimize media content delivery, CloudFront becomes a vital component, caching the static website reduces loading time for users whether they are in different places all over the world. That is why the website contains heavy media like Photo Album should be used to prioritise quick and produce reliable content delivery, which is beneficial for user-experience.

c) *CloudFront with Cognito*

Functionality: Amazon CloudFront uses CDN to cache images and video for users at the nearest edge location, which makes it easily accessible by decreasing time to load content. In terms of encryption, content is delivered via HTTPS and features, specifically origin access identity OAI and IP restriction, controlling who is able to access the information. AWS Cognito is used to authenticate users and the access management, which enables creators to create the identity for authenticated users, therefore enhancing security and user experience.

Reason of using: Similar to the previous CloudFront, the media files can be optimised through CDN so as to reduce time for loading content to provide the best experience for end-users. However, authorised users have to share their signed URLs to people whom they wish to grant access. These URLs are validated via Cognito to make sure that the data is not available to the public. CloudFront grants permission for operations based on keys and policies. Additionally, if the assets policy blocks assets after a certain period, therefore, access is blocked after time expires, which maintains security and effective access control.

d) AWS API Gateway

Functionality: Amazon API Gateway is a full-ranged service that enables developers to design, deploy, maintain, monitor, publicise and secure APIs at any size. It supports the connection between REST and WebSocket APIs for real-time in two-way communication. Key features include integration with Lambda for a highly available computing infrastructure and direct access to several AWS services, traffic control, authorisation and access control, monitoring, and API version management. Therefore, this helps users to efficiently create and manage APIs, connect AWS or other online services, and retrieve data in the AWS Cloud.[10],[11]

Reason of using: Our application's back-end logic mostly utilises functions, with API gateway giving different endpoints for user interactions, including uploading media (specifically video and images), displaying user's data and other essential functionalities. Thanks to Amazon Cognito integration, ensure secure access to these APIs, handling authentication via Cognito and providing JWTs (JSON Web Tokens) for user authentication. API gateway automatically scales up, ensuring responsiveness for the Photo Album web application even during peak demand. This leverages AWS infrastructure to provide quick responses to adapt client requests.

e) AWS WAF (Web Application Firewall)

Functionality: It helps protect web applications and APIs from common web exploits and vulnerabilities that can affect availability, compromise security, and consume resources excessively. It allows users to make custom security rules to block common attack patterns, including cross-site scripting (XSS).[12] Integrating seamlessly with AWS CloudFront, WAF provides real-time traffic monitoring, bot control, and flexible rule filters to enhance the application's security and performance.[13]

Reason of using: In our architecture, there should be comprehensive ways to protect us from common web exploits and attacks, including cross-site scripting (XSS), so that is the reason why placing WAF in front of CloudFront ensures that every incoming traffic is detected and filtered by leveraging its robust security capabilities. Therefore, this will protect our static content when accessing the assets (images and videos) from a signed URL.

3) Logic Tier and Data Tier (Upload Photos and Videos)

a) Amazon S3 (Simple Storage Service)

Functionality: Amazon S3 is a service that allows for the storage and protection of photos and videos. It is extremely scalable and created specifically for this purpose. It is widely recognized for its long-lasting nature, strong protection, and capacity to handle increasing demands.[14]

Reason of using: As the user base of the Photo Album application expands, S3 can effortlessly adjust to handle larger quantities of media without requiring manual intervention. Moreover, S3 has an exceptional level of durability, with a reliability rate of 99.999999999%. It also incorporates strong security measures to ensure the safe storage of customers' photographs and movies.

b) SQS (Simple Queue Service)

Functionality: Amazon SQS is a comprehensive and controlled messaging queuing service that allows for the separation and expansion of microservices, distributed systems, and serverless applications. It facilitates communication between S3 and AWS Lambda (or other processing services) throughout the media processing operation. [15]

Reason of using: SQS separates the media upload process from the processing activities, guaranteeing that the application can manage sudden increases in media uploads without overwhelming the processing services. Moreover, SQS has the capacity to automatically handle enormous volumes of messages, making it crucial for accommodating the expected development of the application.

c) AWS Lambda

Functionality: AWS Lambda is a serverless compute service that runs code in response to events and automatically manages the underlying compute resources. It supports a variety of programming languages and scales automatically based on the number of incoming requests.[16]

Reason of using: Serverless architecture simplifies server management and reduces the burden on the development team, enabling them to concentrate on application logic. Additionally, Lambda exhibits automatic scalability, efficiently managing fluctuating workloads of media processing jobs. Moreover, Lambda's pricing approach, based on pay-as-you-go, guarantees that expenses are only incurred when code is executed. This makes it a cost-efficient option for workloads that occur intermittently.

d) Amazon Rekognition

Functionality: Amazon Rekognition is a service that enhances apps by providing them with the ability to analyze images and videos. The system has the capability to identify and recognize objects, sceneries, and faces in both photos and videos, which makes it highly valuable for a wide range of applications, such as security and content analysis. [17]

Reason of using: Amazon Rekognition automatically tagging and categorizing photos enhances the user experience by making it easier to search and organize media, it implements sophisticated AI functionalities into the application.

e) DynamoDB

Functionality: Amazon DynamoDB is a fully managed NoSQL database service that offers low latency and high performance at any scale. It supports both key-value and document data models and is designed for high availability and durability. [18]

Reason of using: Dynamo DB offers rapid and reliable performance, crucial for storing and retrieving metadata linked to media assets. It automatically adjusts its capacity to accommodate the increasing volume of metadata as the number of users of the application grows. Furthermore, DynamoDB's on-demand capacity option enables efficient management of fluctuating workloads while keeping costs down.

f) Elemental MediaConvert

Functionality: AWS Elemental MediaConvert is a file-based video transcoding service with broadcast-grade features. It enables the creation of video-on-demand (VOD) content for delivery to various devices, providing a range of video formats and resolutions.[19]

Reason of using: AWS Elemental MediaConvert offers extensive compatibility with input and output formats, enabling the conversion of videos uploaded in diverse formats to universally accepted formats that are compatible with a variety of devices. Moreover, it is also capable of efficiently processing large quantities of video transcoding tasks, which is essential for the application's capacity to expand. Compared with Amazon Elastic Transcoder, AWS Elemental MediaConvert is much more advanced and cost-effective.

III. UML Sequence Diagrams and Processing Design

The following sections outline the sequence diagrams that illustrate the interactions between users and services. Additionally, the step-by-step process of the communication will be mentioned in detail with the implementation of 3 main designs: *Security and Monitoring*, Media Uploader and Get Media.

A. Security and Environment Monitoring Processing

The UML diagrams below contain the labels that reside on the lines that simplify the interactions between services:

1.1. The user logs onto the application that must go through the AWS Shield service to make sure that it is not a DDoS attacker.

1.2. This action will also trigger the Lambda function to make the authentication request to AWS Cognito.

1.3. If the user enters the correct credentials, Cognito will verify that and then the user can access the web application successfully.

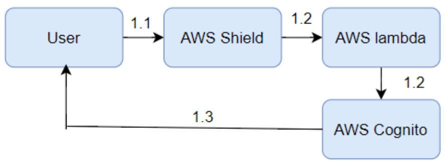


Fig. 2 Authentication Sequence

2.1. When the user tries to access the resource, AWS Shield service verifies if that user is legitimate to prevent the DDoS attack.

2.2. This action will trigger the lambda function to check the authorization of that user through the token generated in Cognito previously.

2.3. Identity information contained in this token will be evaluated by AWS IAM.

2.4. If that user is the authorized user, they are allowed by IAM to have permission to access the resource.

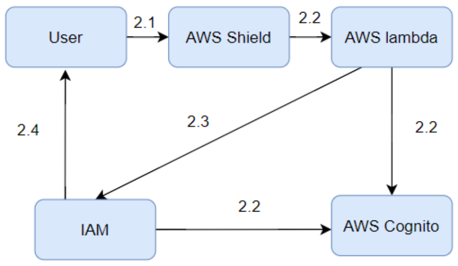


Fig. 3 Authorization sequence

3.1. When the user wants to access the resource in S3 bucket or retrieve data from DynamoDB, it will trigger the Lambda function to request to get this data.
3.2. This request will be processed by AWS KMS. For example, the uploaded data will be encrypted by KMS before being uploaded to S3 bucket.
3.3. Similarly, the data is also encrypted before being stored in DynamoDB.
3.4. KMS ensures that only authorized users can access the encrypted keys

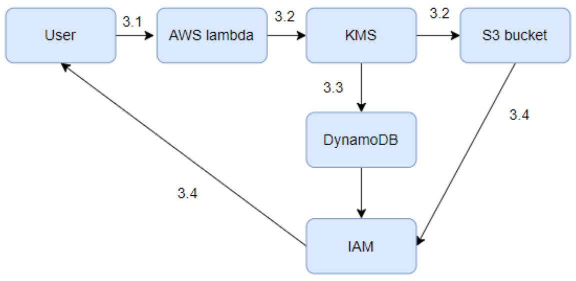


Fig. 4 Data encryption sequence

4.1. AWS CloudTrail monitors API calls and events within the AWS environment by authorization activity.
4.2. AWS CloudTrail monitors API calls and events within the AWS environment by authentication activity.
4.3. AWS CloudTrail monitors API calls and events within the AWS environment by resource upload and access activity.
4.4. When Shield service detects any abnormal events, it will be recorded in Cloud Trail logs.

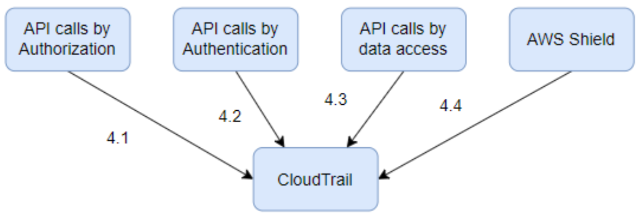


Fig. 5 Auditing sequence

5.1. All CloudTrail logs will be alerted by SNS to the user/administrator
5.2. Unusual activities detected by AWS Shields trigger SNS then SNS will send the notifications to the user/administrator
5.3. Authentication, Authorization and data access activities from a specific user are also alerted by SNS.

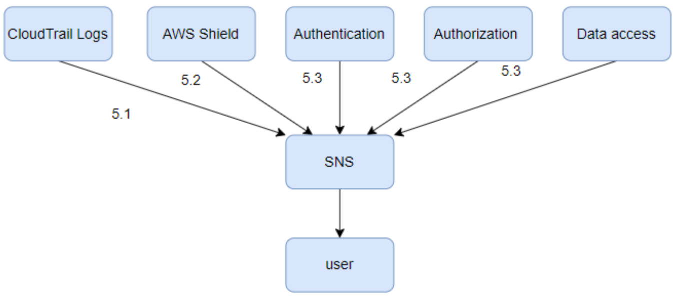


Fig. 6 Elerting notification triggering sequence

B. *Monitoring Processing (Tran Anh Thu Pham)*

Step 23: AWS CloudWatch collects metrics from AWS resources and services

Step 24: CloudWatch alarms monitor metric values and trigger notifications when predefined thresholds are exceeded.

To be more specific, the UML diagram below contains the labels that reside on the lines that simplify the interactions between services:

6.1. AWS CloudWatch collects metrics from the S3 bucket, DynamoDB, Shield services, KMS, IAM, and Cognito service.

6.2. CloudWatch alarms monitor metric values and trigger notifications using SNS service when predefined thresholds are exceeded.

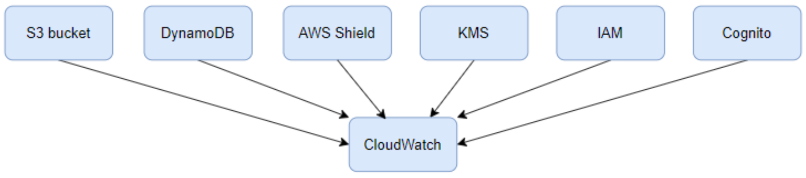


Fig. 7 Monitoring sequence

Referring to the architecture diagram above, There are two different types of monitoring the environment including AWS CloudTrail and AWS CloudWatch. The difference between these two services is that CloudTrail will alert the user/administrator when there is a problem related to the user's security and CloudWatch will alert the user when there is an operational problem occurs.

C. *Request Processing and Request Signed URL*

a. *Request Processing by Authorisation*

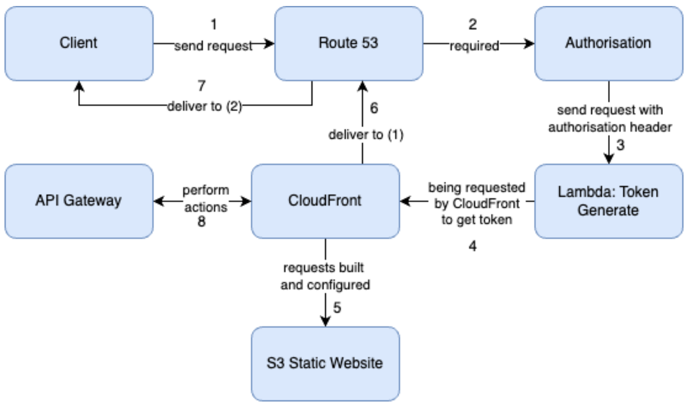


Fig. 8: Authorising access sequence

Step 1: Client users send request to the application web domain via Route 53

Step 2: Require authorisation via login or register in order to send the HTTP/HTTPS request with Authorization header

Step 3: Lambda for generating access token from successful Authorisation header.

Step 4: CloudFront requests the Lambda token function to get value.
Step 5: If successful, CloudFront requests a static web application built and configured at S3 bucket.
Step 6 + 7: Two times deliver a static webpage from Route S3 and then to the end-users.
Step 8: Authorised users can upload and retrieve material from the webpage.

b. Request Signed URL by generate

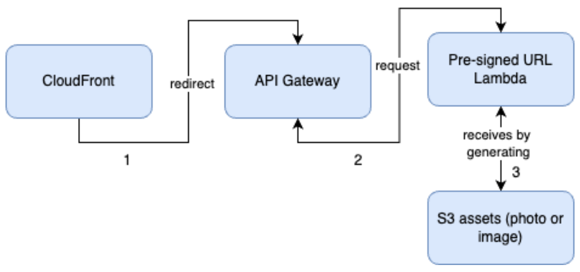


Fig. 9: Unlisted URL request

Step 1: CloudFront is redirected to API Gateway to request a pre-signed URL
Step 2: API Gateway receives the request for a pre-signed URL and forwards it to a Lambda function
Step 3: Lambda function generates the pre-signed URL for the private content, which is triggered by the authorised user, and then retrieves the content from the S3 bucket if the URL is valid and not expired.

D. Access Signed URL

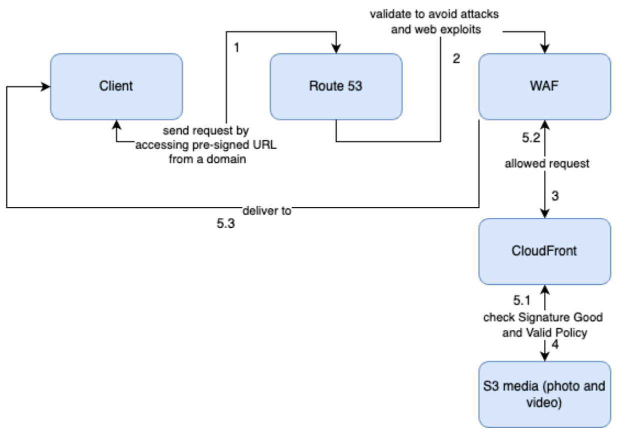

validate to avoid attacks and web exploits
1
2
Client
Route 53
WAF
send request by accessing pre-signed URL from a domain
5.2
allowed request
3
deliver to
5.3
CloudFront
5.1
check Signature Good and Valid Policy
4
S3 media (photo and video)

Fig. 10: Unlisted URL access

Step 1: Client users send a request to the application web domain via Route 53 with their signed URL.
Step 2: To display media via a signed URL, the code cookie's value is sent over through the CloudFront distribution endpoint, but the AWS WAF must assess if the cookie should be enabled to trigger the Lambda function URL origin.
Step 3: Allowed requests are directed to the endpoint of the CloudFront distribution.
Step 4: The URL signature is allowed to retrieve the data from S3.
Step 5.1 -> 5.2 -> 5.3: The webpage returns the S3 bucket as displayed successfully or denied access if the signature URL is invalid.

E. Upload Photos and Videos Processing

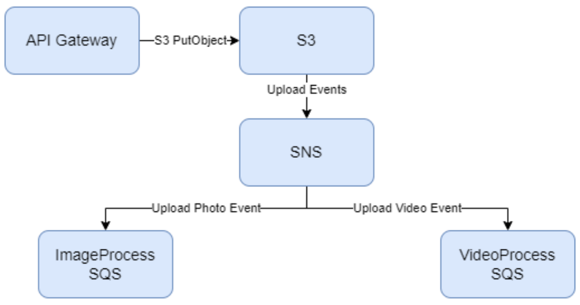


Fig. 11: Upload media files process

Step 1: Media (pictures and videos) are uploaded by users through the API Gateway, which serves as the point of entry for the upload requests. The uploaded media is stored in the S3 bucket.
Step 3: When a new media file is uploaded to S3, it triggers an event, which sends a message to SNS.
Step 4: The SNS distributes the notification to the appropriate SQS queues based on the media type (specified in the message).

a. If the uploaded media is an image, the notification is sent to the ImageProcess SQS queue:

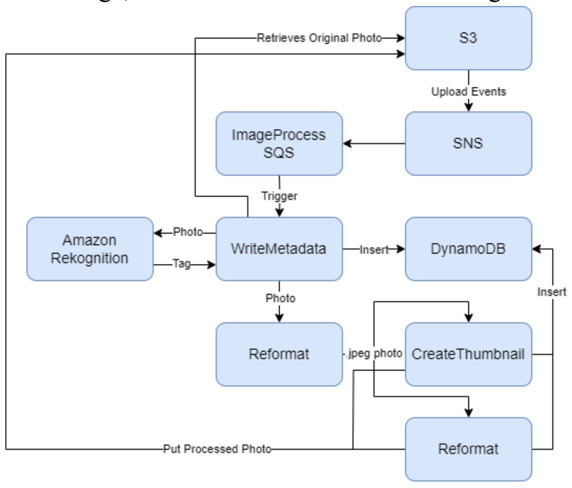


Fig. 12: Image Processing Sequence

Step a.1: The ImageProcess SQS triggers the WriteMetadata Lambda function: First, it will retrieve the original photo from S3, and then the photo will go through Amazon Rekognition to be automatically tagged. Finally, the metadata (including the tag) of the original photo will be inserted into DynamoDB.
Step a.2: The photo will then be sent to the Reformat function, where it's formatted to a .jpeg photo, a standard web image format.
Step a.3: CreateThumbnail function will create a thumbnail function based on the formatted photo, which metadata will be inserted into DynamoDB.
Step a.4: Reformat function will create various resolutions of the image suitable for different devices, which metadata will be inserted into DynamoDB.
Step a.5: All versions created (thumbnails and various resolutions) will be added to the S3 bucket.

b. If the uploaded media is a video, the notification is sent to the VideoProcess SQS queue:

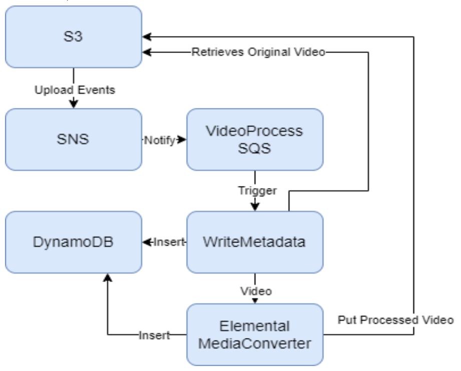


Fig. 13: Video Processing Sequence

Step b.1: The VideoProcess SQS triggers the WriteMetadata Lambda function: First, it will retrieve the original video from S3, and then the metadata of the original video will be inserted to DynamoDB.
Step b.2: The video will then be sent to the AWS Elemental MediaConvert for transcoding into different resolutions and formats.
Step b.3: Then the metadata of the processed video will be updated in the DynamoDB
Step b.4: The processed video is put into the S3 bucket.

IV. Design Rationale

A. Scalability

The AWS photo Album application leverages a serverless architecture to provide a scalable solution for storing, processing, and retrieving photos and videos:

1. **IAM:** IAM handles the scaling of user management. To be more specific, IAM can manage the growing number of access policies when the number of users increases. [1]
2. **Cognito:** Cognito scales automatically with the number of users and the authentication requests. It supports millions of users and can handle multiple sign-in and sign-up requests at the same time. [2]
3. **Route 53:** Route 53 can handle large volumes of DNS queries and scales automatically to manage high traffic loads. [7]
4. **CloudFront:** CloudFront uses a global network of edge locations to cache and serve content. It scales automatically to handle high volumes of requests, reducing the load on the origin servers and providing low-latency content delivery. [8]
5. **API Gateway:** API gateway scales automatically to handle increased API request volumes. It can process many API calls, integrating with other services such as Lambda or Cognito. [11]
6. **SQS:** SQS decouples the media upload process from the processing pipeline. It scales automatically to handle an increasing number of messages, ensuring that no data is lost. [15]
7. **AWS Lambda:** Lambda functions scale automatically based on the number of incoming requests. As there are no servers to manage, Lambda can handle many executions for the unpredictable traffic load. [16]

B. Reliability

1. **Route 53:** Route 53 is a highly reliable DNS service. It offers features such as health checks and failover routing to ensure that user traffic is always directed to healthy checkpoints. [7]
2. **CloudFront:** CloudFront improves reliability by caching content at edge locations worldwide. This reduces the load on the origin servers and provides multiple layers of redundancy, ensuring content delivery is always available if some edge locations have issues. [8]
3. **CloudTrail and CloudWatch**: CloudTrails logs all API calls and CloudWatch monitors resources, both providing immediate alerting messages. This ensures any issues can be quickly identified and addressed to maintain the reliability of the Photo Album application.
4. **AWS Shield:** Shield provides continuous DDoS protection, ensuring the application services remain available even under attack. Shield Standard is automatically available to protect against the most common network and transport layer DDoS attacks. [4]
5. **SQS:** SQS ensures reliable message delivery. It can automatically handle message duplication to ensure messages are processed at least once. [15]
6. **IAM:** IAM provides reliable access management to make sure that permissions and policies are consistently enforced. This reduces the risk of unauthorized access and disruption. [1]

C. Security

1. **IAM**: IAM manages access to AWS resources by defining specific users, groups, and roles that can access specific resources and under specific conditions. IAM policies enforce the least privilege and mitigate over permission. [1]
2. **Cognito**: Cognito provides user sign-in, sign-up, and access control, user authentication through its user directory. Cognito issues OAuth 2.0 tokens for user sessions ensuring secure access to the application and its resources. [2]
3. **Simple Notification Service (SNS)**: SNS ensures secure communication between publishers and subscribers. SNS topics can be restricted to specific IAM roles to prevent unauthorized access. [5]
4. **KMS**: KMS manages cryptographic keys, enabling data encryption and decryption. [3]
5. **CloudTrail**: CloudTrail provides a comprehensive audit trail of all API calls, helping detect unauthorized access and other security incidents. [3]
6. **AWS Shield**: Shield Standard automatically defends against most common network and transport layer DDoS attacks. Shield Advances offers additional protections. [4]
7. **CloudWatch**: CloudWatch monitors AWS resources and applications. CloudWatch triggers alerts when there are security incidents. [6]

D. Cost Effective

DynamoDB is a fully managed NoSQL with unmatched scalability and speed, making it ideal for Photo Album applications, compared with RDS, a relational database with higher cost and less capacity due to vertical scaling, and uses more computer resources. [18]

Serverless Architecture: AWS Lambda and other serverless services reduce the management of servers, decreasing operational costs and offering pay-as-you-go pricing for compute time usage.

Scalable Solutions: API Gateway and Lambda automatically scale up on-demand, minimising unnecessary expenses.

Optimized Storage: Amazon S3 provides cost-efficient, scalable storage measures. [14]

Event-Driven Processing: S3 bucket and SQS for media processing reducing continuous costs for operations.

Integrated Security: AWS WAF and Shield provide strong security without allocating more infrastructure.

E. Global Response Time Improvement

By using Route 53 alongside CloudFront, the global response time can be improved effectively at the photo album's application, with a content delivery network that stores content in edge locations globally at low latency. Therefore, this enhances website and application efficiency, allowing quick delivery of asset files.

Route 53 is also a highly flexible and dependable DNS web platform that converts IP addresses from domain names. At its current architecture, Route 53 delivers internet traffic through the closest CloudFront edge point to make websites and applications accessible via domain names.[20]

F. Video Handling and Media Processing

The media processing architecture for the Photo Album application is designed to efficiently handle an increasing number of media uploads, while also prioritizing scalability, reliability, and improved user experience. The solution is based on the following fundamental design principles:

1) Event-Driven architecture

The event-driven architecture enables the system to promptly react to modifications or occurrences (such as the uploading of material) in real-time, instantly initiating the required processing workflows.

Scalability: The system's ability to manage a larger number of media uploads is ensured through event-driven processing, eliminating the need for manual intervention. When a new photo or video is uploaded to Amazon S3, it immediately generates an event that starts the media processing.

Cost-effectiveness is achieved by utilizing resources only when events occur, which ensures that operational expenses are in line with real consumption and prevents wasteful expenditure.

The architecture is designed to be flexible, allowing different processing tasks to be executed on the most suitable platform:

Lambda functions are well-suited for doing lightweight and short-duration operations such as picture resizing, thumbnail creation, and metadata extraction. [16]

Elemental MediaConvert is a specialized tool designed for video processing. It offers extensive capabilities and optimizations specifically for managing video files. [19]

DynamoDB is a NoSQL database that is completely controlled and offers fast response times and excellent performance. It is particularly useful for efficiently storing and retrieving metadata. DynamoDB's capacity to scale and adapt makes it highly effective in managing the dynamic aspects of storing and retrieving metadata. [18]

This architecture also promotes decoupling, ensuring that each part can scale independently and operate without being tightly integrated with others, promoting robustness and flexibility:

Loose Coupling: Utilizing services such as SNS and SQS, which enables the separation of the upload process from the processing. In the event of a single point of failure, there is no direct impact on the other components.

Simplified Maintenance: Independent components can be updated, scaled, or replaced without requiring a complete system overhaul, simplifying maintenance and upgrades.

2) AI Integration

Integrating AI into media processing enhances the application's functionality by automating tasks that would otherwise require significant manual effort. With AWS Rekognition's integration, the media file tags can be automatically identified, setting the stage for enhanced user experiences and capabilities in the future. [17]

V. Alternative Solution

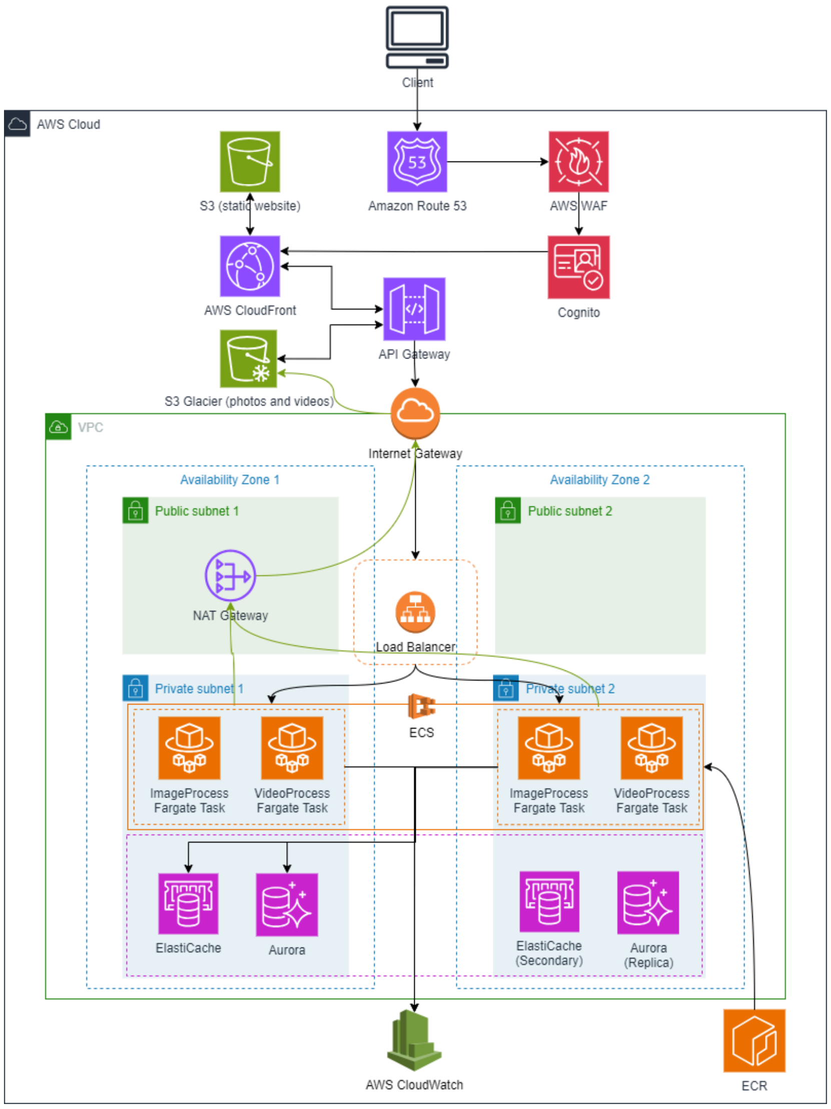


Fig. 14: Alternative solution diagram

This architecture is built upon the concept of containerization, utilizing the power of containers to encapsulate the application code along with its dependencies.

Key components of the alternative solution:

Amazon S3 Glacier: This service offers secure, long-lasting, and easily expandable storage for data archives. It guarantees the reliable and cost-efficient preservation of data for lengthy periods. [14]

Amazon VPC (Virtual Private Cloud): VPC establishes a virtual network environment in the cloud that is isolated from other virtual networks. It improves security and allows for a personalized network setup. [21]

NAT Gateway: Facilitating outbound traffic from private subnets to the internet, NAT Gateway enhances security by acting as a gateway for instances in private subnets, allowing them to access the internet. [21]

Amazon Elastic Container Service (ECS): ECS serves as the orchestration engine for deploying, managing, and scaling containerized applications. It automates the deployment and scaling of the containers while maintaining high availability. [22]

AWS Fargate: Fargate serves as the fundamental framework of this architecture. It dynamically allocates computational resources, optimizing the consumption of resources and reducing expenses. [22]

ElastiCache: ElastiCache enhances the performance of the application by providing an in-memory caching layer. It enables the application to retrieve data quickly by storing frequently accessed data in memory, thereby reducing latency and improving overall responsiveness. [23]

Amazon Aurora: As a fully managed relational database service, Aurora offers high performance, scalability, and availability for media data. It ensures durability and reliability while handling the heavy lifting of database management tasks. [24]

A. Comparison

Database: DynamoDB vs Amazon Aurora

Performance and Scalability: DynamoDB has low latency, seamless scalability, and cost-effectiveness for recording metadata with simple table structure by providing single-digit millisecond latency [18], while Aurora also has high performance, scalability, and reliability for relational databases, it requires proper configuration with minimum capacity and only suitable for relational database structure. [24]

Cost-Effectiveness: DynamoDB is a pay-as-you-go pricing model, while Aurora might lead to needless capacity used.

Based on the above criteria comparison, DynamoDB is a better choice for the Photo Album.

Storage: S3 vs S3 Glacier

S3 offers highly available and durable object storage with low latency access, suitable for frequently accessed data and real-time processing, while S3 Glacier offers cost-effective archival storage for long-term data retention, ideal for infrequently accessed data or compliance requirements [14]. If the album photo website is made to access photos and videos frequently, S3 is a suitable choice but if the album photo website is made for users to store their photos and videos online, S3 Glacier can be used in terms of cost-effectiveness, and suitability.

Computing resources: AWS Lambda vs AWS Fargate

Cost-Effectiveness: AWS Lambda is a pay-per-use model (pay for the compute time) [16] while AWS Fargate is more expensive in media processing as it pays for the utilized resources. [22]

Scalability: AWS Lambda scales automatically based on the volume of incoming events [16] while AWS Fargate requires ECS service with Amazon CloudWatch for monitoring (which also costs more than AWS Lambda)[22].

Reliability: AWS Lambda ensures reliability with built-in fault tolerance and automatic retries for failed executions [16], while AWS Fargate's reliability might vary based on the underlying infrastructure [22].

Based on the above criteria comparison with the photo album requirements, AWS Lambda is a better choice in terms of scalability, reliability, and cost-effectiveness.

Cache Options For Media Processing: ElasticCache vs No cache options.

ElasticCache enhances media processing through in-memory caching, reducing latency and improving responsiveness [23]. However it also introduces more cost, so we consider not to include it into our solution as the media processing speed is not a concern.

Overall Comparison: Container-based Architecture vs Serverless Architecture

Security: Our serverless solution has included a comprehensive and complex authorization and tokens process so that only a user can access their own media files while the alternative solution doesn't contain this functionality.

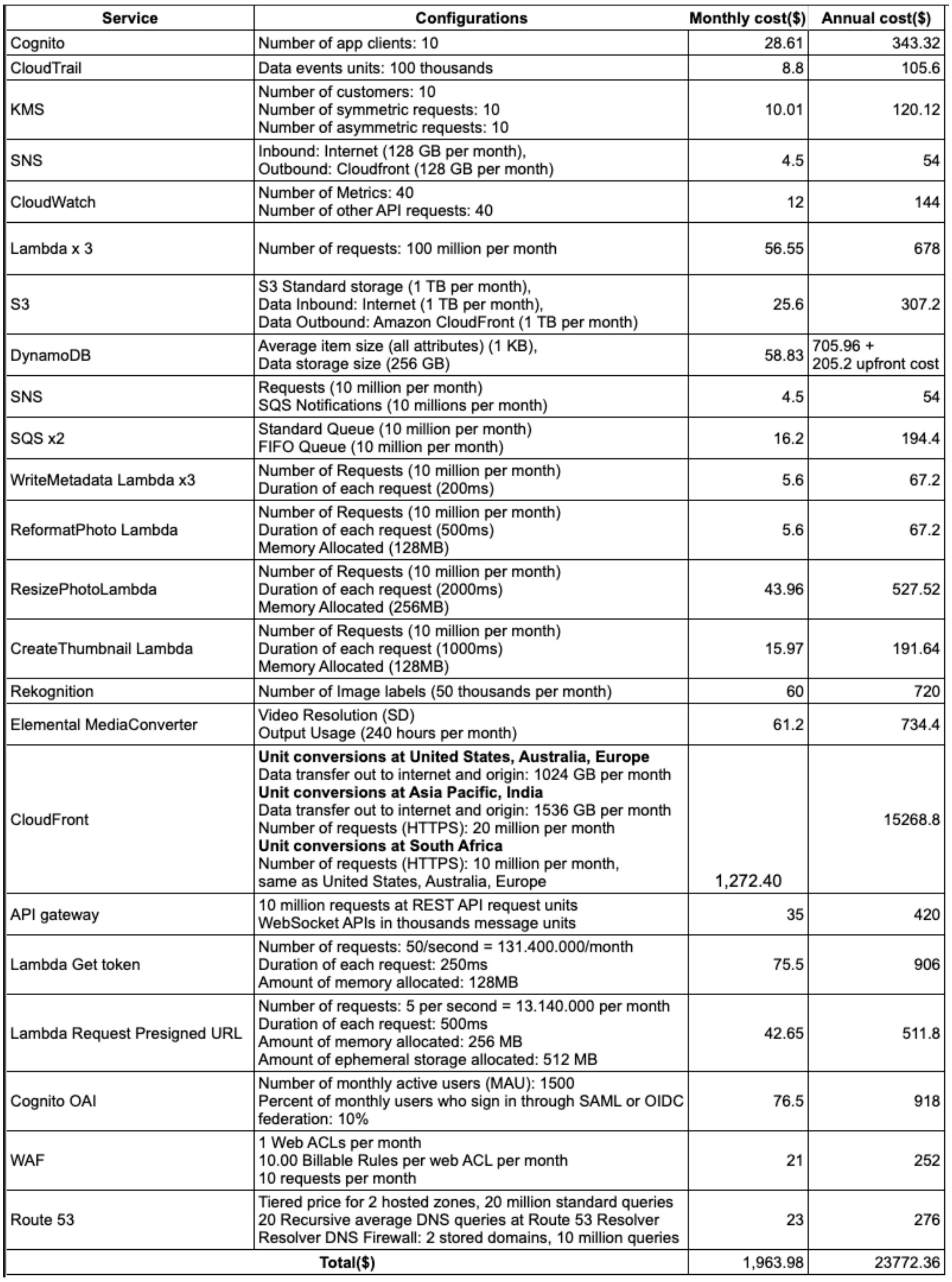
Scalability: Based on the above comparison between components, both architectures offer high scalability but components like Lambda, SQS, and DynamoDB offer seamless scalability and flexibility than their counterparts.

Reliability: Serverless components like Lambda and SQS abstract away infrastructure management tasks, reducing operational overhead and enabling developers to focus on application logic rather than infrastructure while their counterparts require proper configuration for reliability.

Cost-Effectiveness: Pay-per-use pricing models for Lambda and DynamoDB make event-driven architecture more cost-effective, especially for sporadic workloads or applications with unpredictable usage patterns while their counterparts may waste budget on low-traffic occasions. Moreover, Container-based Architecture also involves NAT Gateway and VPC utilization, which costs much more than serverless architecture.

Overall, Event-Driven Architecture is a more suitable architecture for the photo album website.

VI. Cost Estimation



Service	Configurations	Monthly cost($)	Annual cost($)
Cognito	Number of app clients: 10	28.61	343.32
CloudTrail	Data events units: 100 thousands	8.8	105.6
KMS	Number of customers: 10 Number of symmetric requests: 10 Number of asymmetric requests: 10	10.01	120.12
SNS	Inbound: Internet (128 GB per month), Outbound: Cloudfront (128 GB per month)	4.5	54
CloudWatch	Number of Metrics: 40 Number of other API requests: 40	12	144
Lambda x 3	Number of requests: 100 million per month	56.55	678
S3	S3 Standard storage (1 TB per month), Data Inbound: Internet (1 TB per month), Data Outbound: Amazon CloudFront (1 TB per month)	25.6	307.2
DynamoDB	Average item size (all attributes) (1 KB), Data storage size (256 GB)	58.83	705.96 + 205.2 upfront cost
SNS	Requests (10 million per month) SQS Notifications (10 millions per month)	4.5	54
SQS x2	Standard Queue (10 million per month) FIFO Queue (10 million per month)	16.2	194.4
WriteMetadata Lambda x3	Number of Requests (10 million per month) Duration of each request (200ms)	5.6	67.2
ReformatPhoto Lambda	Number of Requests (10 million per month) Duration of each request (500ms) Memory Allocated (128MB)	5.6	67.2
ResizePhotoLambda	Number of Requests (10 million per month) Duration of each request (2000ms) Memory Allocated (256MB)	43.96	527.52
CreateThumbnail Lambda	Number of Requests (10 million per month) Duration of each request (1000ms) Memory Allocated (128MB)	15.97	191.64
Rekognition	Number of Image labels (50 thousands per month)	60	720
Elemental MediaConverter	Video Resolution (SD) Output Usage (240 hours per month)	61.2	734.4
CloudFront	**Unit conversions at United States, Australia, Europe** Data transfer out to internet and origin: 1024 GB per month **Unit conversions at Asia Pacific, India** Data transfer out to internet and origin: 1536 GB per month Number of requests (HTTPS): 20 million per month **Unit conversions at South Africa** Number of requests (HTTPS): 10 million per month, same as United States, Australia, Europe	1,272.40	15268.8
API gateway	10 million requests at REST API request units WebSocket APIs in thousands message units	35	420
Lambda Get token	Number of requests: 50/second = 131.400.000/month Duration of each request: 250ms Amount of memory allocated: 128MB	75.5	906
Lambda Request Presigned URL	Number of requests: 5 per second = 13.140.000 per month Duration of each request: 500ms Amount of memory allocated: 256 MB Amount of ephemeral storage allocated: 512 MB	42.65	511.8
Cognito OAI	Number of monthly active users (MAU): 1500 Percent of monthly users who sign in through SAML or OIDC federation: 10%	76.5	918
WAF	1 Web ACLs per month 10.00 Billable Rules per web ACL per month 10 requests per month	21	252
Route 53	Tiered price for 2 hosted zones, 20 million standard queries 20 Recursive average DNS queries at Route 53 Resolver Resolver DNS Firewall: 2 stored domains, 10 million queries	23	276
Total($)		1,963.98	23772.36

Fig. 15: Cost estimation for proposed solution

VII. Conclusion

The AWS Photo Album Application architecture outlined in this report demonstrates a scalable, reliable, secure, and cost-effective solution implemented in various AWS services. Scalability is achieved through the automatic scaling capabilities of AWS Lambda, API Gateway, CloudFront, etc. This ensures that the application remains responsive and efficient regardless of user traffic volumes. Reliability is achieved by the use of AWS services such as Route 53, CloudFront, SQS, etc. Security is achieved by using IAM, Cognito, KMS, AWS Shield, etc to manage user access, authentication, data encryption, and key management. In Summary, the proposal for serverless cloud architecture for the Photo Album application is a comprehensive solution that balances scalability, reliability, security, and cost-effectiveness. This architecture not only meets the business requirements but is also available for future enhancement and scalability.

References

[1] AWS, "Understanding How IAM Works - AWS Identity and Access Management," *Amazon.com*, 2019. https://docs.aws.amazon.com/IAM/latest/UserGuide/intro-structure.html

[2] AWS, "What Is Amazon Cognito? - Amazon Cognito," *AWS Documentation*. https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

[3] AWS, "Features | AWS Key Management Service (KMS) | Amazon Web Services (AWS)," *Amazon Web Services, Inc*. https://aws.amazon.com/kms/features/

[4] AWS, "How AWS Shield Works - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced," *Amazon.com*, 2015. https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html

[5] AWS, "What is Amazon SNS? - Amazon Simple Notification Service," *AWS Documentation*. https://docs.aws.amazon.com/sns/latest/dg/welcome.html

[6] AWS, "How Amazon CloudWatch Works - Amazon CloudWatch," *AWS Documentation*. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_architecture.html

[7] AWS, "Amazon Route 53 - Amazon Web Services," *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/route53/

[8] AWS, "What Is Amazon CloudFront? - Amazon CloudFront," *Amazon.com*, 2016. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

[9] AWS CloudFront, "Content Delivery Network (CDN) | Low Latency, High Transfer Speeds, Video Streaming | Amazon CloudFront," *Amazon Web Services, Inc.*, 2018. https://aws.amazon.com/cloudfront/

[10] AWS, "Amazon API Gateway," *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/api-gateway/

[11] AWS, "What Is Amazon API Gateway? - Amazon API Gateway," *Amazon.com*, 2019. https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

[12] AWS, "AWS WAF - Web Application Firewall - Amazon Web Services (AWS)," *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/waf/

[13] AWS, "Using AWS WAF to Control Access to Your Content - Amazon CloudFront," *docs.aws.amazon.com*. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-web-awswaf.html

[14] AWS, "What is Amazon S3? - Amazon Simple Storage Service.", *docs.aws.amazon.com*. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[15] AWS, "What is Amazon Simple Queue Service? - Amazon Simple Queue Service." , *docs.aws.amazon.com*. https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

[16] AWS, "What is AWS Lambda? - AWS Lambda.", *docs.aws.amazon.com*. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[17] AWS, "Amazon Rekognition (AMS SSPS) - AMS Advanced User Guide.", *docs.aws.amazon.com*. https://docs.aws.amazon.com/managedservices/latest/userguide/rekognition.html

[18] AWS, "What is Amazon DynamoDB? - Amazon DynamoDB." , *docs.aws.amazon.com*. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

[19] AWS, "AWS Elemental MediaConvert (AMS SSPS) - AMS Advanced User Guide." , *docs.aws.amazon.com*. https://docs.aws.amazon.com/managedservices/latest/userguide/amz-elemental-media-convert.html

[20] AWS, "Routing traffic to an Amazon CloudFront distribution by using your domain name - Amazon Route 53," *docs.aws.amazon.com*. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-cloudfront-distribution.html

[21] AWS, "What is Amazon VPC? - Amazon Virtual Private Cloud." *docs.aws.amazon.com*. https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

[22] AWS, "Amazon ECS on AWS Fargate - Amazon Elastic Container Service.", *docs.aws.amazon.com*. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS_Fargate.html

[23] AWS, "Redis and Memcached-Compatible Cache – Amazon ElastiCache – Amazon Web Services," Amazon Web Services, Inc. https://aws.amazon.com/elasticache/

[24] AWS, "High availability for Amazon Aurora - Amazon Aurora.", *docs.aws.amazon.com*. https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.AuroraHighAvailability.html