Serverless/Event Driven Architectural Design Project

Taliah Brooker[1], Thai Son Nguyen[2], Vu Duc Tran[3], Muygech Lim[4]

Swinburne University of Technology, Australia

[1]104017813@student.swin.edu.au; [2]104510464@student.swin.edu.au;
[3]104175614@student.swin.edu.au; [4]102601256@student.swin.edu.au

Abstract: This report presents a serverless/event driven solution for a scalable highly available web site using AWS services and architecture. The objective is to meet increasing demand following the success of the Photo Album Application, and improve the architecture design, with a focus on cost effectiveness, scalability and improving efficiency. The proposed solution in this report is designed to meet the requirements as outlined in the business scenario and provides an outline of the architecture and justifications for the choices made, as well as some alternative options.

Keywords: cloud services, AWS, web services, architecture design, serverless

I. INTRODUCTION

The Photo Uploader Web Application has been met with success and needs to meet increasing user demand. This report outlines improved architecture design and analysis of possible AWS services to be used to create a scalable and serverless application. Several key requirements have been identified by the business that have been followed in creating a possible solution. Some of the key requirements are as follows: the application should be serverless/event-driven, managed services should be used where possible, including storing photos in an S3 bucket, the solution should be scalable to meet projected future demand of doubling every 6 months for the next 2-3 years and the solution should be cost-effective and secure. An analysis of the chosen solution and of alternatives that have been explored are outlined as well.

II. ARCHITECTURE DIAGRAM

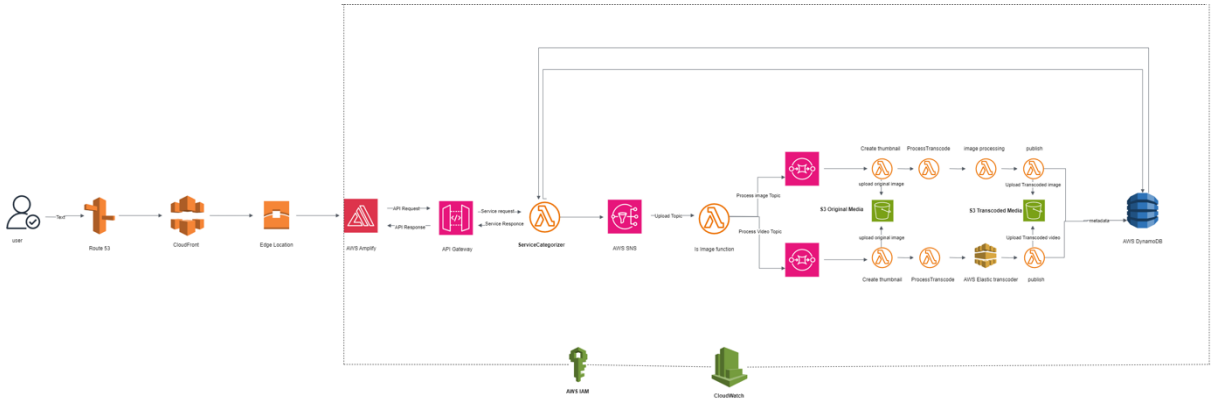

Figure 1 - Diagram of AWS services

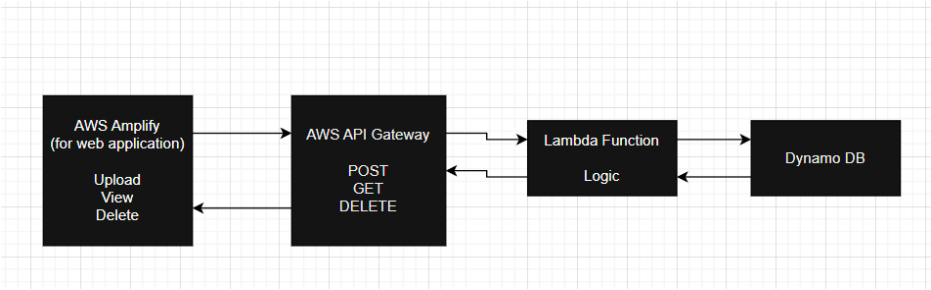

Figure 2 – Business logic diagram

III. COLLABORATION DIAGRAMS

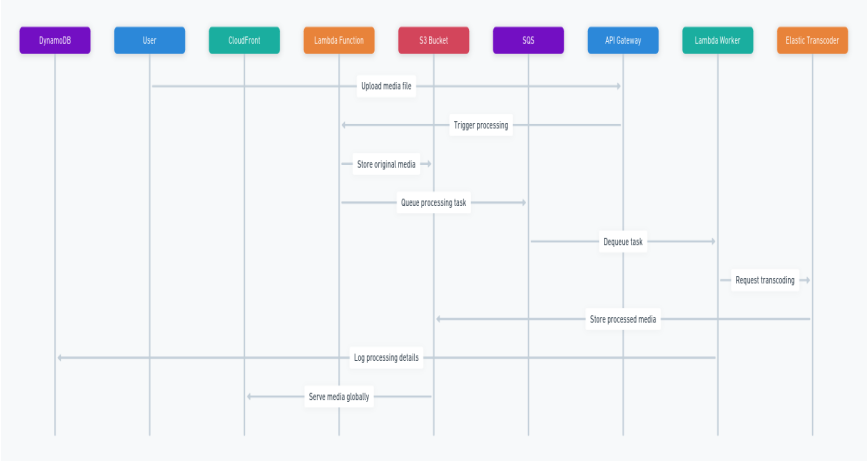

Figure 3

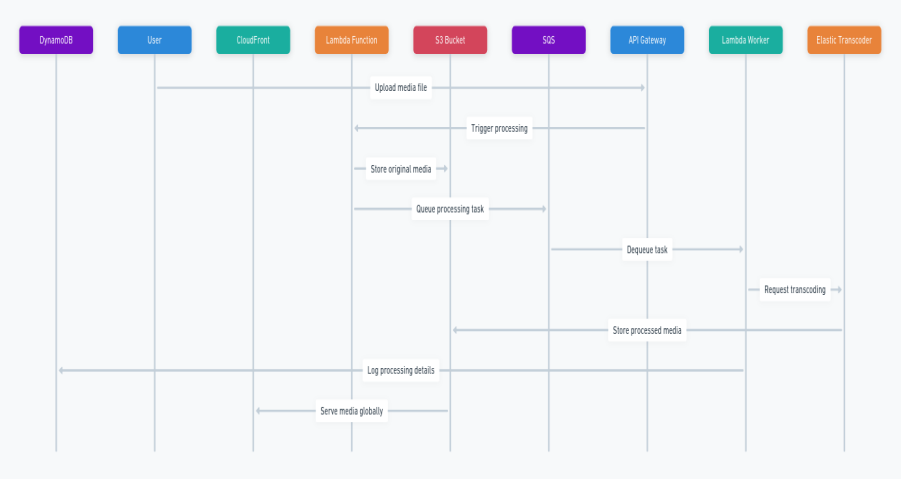

Figure 4

IV. DESCRIPTION OF SERVICES USED

- ***AWS S3 (Simple Storage Service):*** It stands as an object storage solution, renowned for its unmatched scalability, data accessibility, security, and operational efficiency.

- ***AWS Lambda:*** This service revolutionizes computing by allowing code execution devoid of server provisioning or management burdens.
- ***AWS CloudFront:*** Crafted for web content acceleration, it swiftly delivers both static and dynamic elements like .html, .css, .js, and images directly to users.
- ***AWS Route 53:*** The pinnacle of DNS web services, providing high availability and scalability alongside domain registration, DNS routing, and health checking capabilities.
- ***AWS DynamoDB:*** As a fully managed NoSQL database service, it guarantees rapid and consistent performance with seamless scalability.
- ***AWS Amplify:*** Empowering front-end web and mobile developers, it offers a comprehensive suite of tools and services to create secure and scalable full stack applications, harnessing the prowess of AWS.
- ***AWS SQS (Simple Queue Service):*** *This* messaging queue service facilitates seamless communication and storage of messages among software components.
- ***AWS Cognito:*** A managed service that streamlines user authentication, authorization, and management, facilitating swift integration of user-centric functionalities into web and mobile applications.
- ***AWS API Gateway:*** Serving as a cornerstone for API management, it enables effortless creation, publication, maintenance, monitoring, and securing of REST, HTTP, and WebSocket APIs at any scale.
- ***AWS WAF (Web Application Firewall):*** A crucial guardian of web applications, it monitors end-user web requests and manages content access to bolster security measures.

V. DESIGN RATIONALE

A. *PERFORMANCE*

CRITERIA	SOLUTION
FAST MEDIA UPLOAD AND DOWNLOAD SPEEDS.	Utilize CloudFront to distribute material to edge locations for quicker user delivery and to cache the website's static content.
EFFICIENT STORAGE AND RETRIEVAL OF PHOTOS.	Based on the user's location, use Route 53 to route traffic to the closest CloudFront edge location.
Even with high traffic and big photo files, or when users are from countries other than Australia, the website is expected to load fast, effectively, and with the least amount of delay possible.	Use S3 Transfer Acceleration, a feature of S3, to speed up content transfers to and from S3 by up to 500% when transferring large things over long distances. USE CLOUDWATCH TO MONITOR THE PERFORMANCE OF THE WEBSITE, INCLUDING LATENCY, ERROR RATES, AND RESOURCE USAGE. Use CloudWatch to monitor the performance of the website, including latency, error rates, and resource usage.

Table 1: Performance Solutions

B. *SCALABILITY*

Criteria	Solution
The website must be capable of handling growing traffic and storage demands, efficiently distributing workload across multiple servers or regions. It should also be able to automatically adjust resources based on fluctuating traffic patterns.	Utilize CloudFront to scale the delivery of content to users worldwide and manage spikes in traffic effectively.
The architecture should be reusable and adaptable, facilitating future modifications such as the addition of new features (e.g adding an AI to identify tag of media)	Use API GATEWAY to create RESTful API will later be used for the website's backend, handle API request
	AWS Amplify is used to create full-stack websites including frontend and backend Use DynamoDB to scale the metadata storage and retrieval Use S3 to store and retrieve content AWS SNS and SQS for notification and queueing to handle the huge amount of number of requests This design system is designed to be decoupling enough that help the future's modification become flexible, for example: adding a new lambda function to do some business logic in the future

Table 2: Scalability Solutions

C. *RELIABILITY*

Criteria	Solution
The website is highly available and able to recover quickly from failures (regular backups to prevent data loss).	Use Amazon S3 for durable and highly available storage. For high availability, use DynamoDB to replicate data across several regions (prevent data loss). For a NoSQL database that is both scalable and highly available, use DynamoDB. Use CloudFront to handle failovers and deliver content from the nearest accessible edge location.

Table 3: Reliability Solutions

D. *SECURITY*

Criteria	Solution
User data is protected and can not be accessed by a third party (eg: amazon,...)	Use AWS IAM for access control and permissions management. Use Amazon S3 for secure object storage with an appropriate permission script.

The website is protected from unauthorized access, theft, or destruction of data.	Use CloudWatch to monitor and detect security threats and anomalies. Use API Gateway to implement permission and authentication for the backend API.

Table 4: Security Solutions

E. *COST*

Based on the assignment requirements, there has been a doubling in demand for the application every six months, and that trend is expected to continue during the ensuing two or three years. To project future website operating costs, the following cost summaries for total media upload sizes at 50GB, 100GB, and 200GB are displayed:
DynamoDB calculated cost bast on MB
Each media uploaded has an assumed size of 150kb

Services	Cost ($)	Total ($)
DynamoDB	Data storage: 12.5 Write setting: 65.5 Read setting: 3.27	75
Route 53	Hosted zone: 0.5	0.5
CloudFront	Data Transfer Out: 5.7 Upload via HTTPS request: 0,43	6.1
S3	Standard: 1.15 Data transfer in and out: 4.5	5.65
Amplify	Free for first 12 months	0
API Gateway (First 333 mil)	API call (First 333 mil): 1.22	1.22
SNS free by Lambda Endpoint	0	0
SQS	Data transfer in and out: 4.5 Free queues	4.5
Elastic Transcoder	With a quality of 720p and 300 minutes video, the cost will be : 9	9
Lambda	Request (each media need 4 request) : 0.28	0.28
CloudWatch	Nine Lambda functions are present. Suppose that each of them is connected to nine metrics: 25.2	25.2
Total		127,47$

Table 5: Estimated Costs

F. *JUSTIFICATION*

V.F.1 INFRASTRUCTURE MANAGEMENT

The architecture reduces the requirement for internal system administration and storage by utilizing AWS managed cloud services, including AWS S3, AWS Amplify, AWS CloudFront, AWS DynamoDB, and AWS Lambda.

V.F.2 SCALABILITY

The website grows every 6 months, and the architecture that is suggested above is made to handle the need to handle anticipated growth by implementing an event-driven/serverless architecture. Using different managed cloud services, like AWS Lambda functions, which can automatically scale up

or down dependent on the amount of incoming traffic, is one way to achieve this. AWS DynamoDB as a database solution to make sure that the web can handle numerous requests and have no effect on its performance.
Furthermore, the architecture's event-driven approach, which makes use of AWS Lambda, SNS, and SQS, guarantees that the application components are separated, which will improve the architecture's scalability and dependability.

V.F.3 DATABASE

Choosing DynamoDB instead of AWS RDS is our solution since DynamoDB uses NoSQL which will allow automatic scaling of resources, indicating that the application is flexible and scalable. DynamoDB's serverless architecture will also make it a more affordable option for our design, depending on the application, DynamoDB will charge for the throughput used and the volume of data saved.
The optimum user experience and ease of operation for the business owner will also be guaranteed by the data replication across different availability zones of DynamoDB.

V.F.4 PERFORMANCE

Offshore clients' user experience must be catered to as the website's popularity of service continues to rise both inland and offshore in Australia.
For our company's needs, utilizing AWS CloudFront and AWS Amplify can greatly enhance cross-region performance. Our consumers benefit from reduced latency and faster content delivery thanks to the content delivery network that uses AWS CloudFront to cache and deliver material from the nearest location to the user.

VI. ALTERNATIVE SOLUTIONS COMPARISON

We carefully evaluated various options when designing the architecture for the Photo Album app. We analysed multiple choices using performance data and usage metrics to determine the best solution.

A. *ALTERNATIVE STORAGE SOLUTIONS*

VI.A.1 OPTION 1: KEEPING STORAGE ON-PREMISES

We considered storing our media files on our PCs instead of using AWS. However, as our app gains more users, this could cause significant slowdowns and make it challenging to maintain smooth functionality. Keeping our data accessible and secure would be difficult and costly with our current hardware, and there's also the risk of physical theft. Additionally, AWS provides superior security tools that we can't match. Cost analysis showed that purchasing and maintaining our own equipment would be more expensive than using AWS S3.

VI.A.2 OPTION 2: USING A DIFFERENT CLOUD COMPANY

We also explored using other cloud services. While many companies offer great services, AWS stands out for its global reliability and availability. Other companies have their strengths, but AWS's unique features align well with our project needs. AWS is renowned for its high security, including encryption and access controls. Financially, AWS offers reasonable costs and free usage up to certain limits. Overall, AWS appears to be the best option for us in terms of cost and meeting all our requirements.

B. *Alternative Database Solutions*

VI.B.1 Option 1: Regular Cloud Computers

We considered using standard cloud computers like EC2 instances instead of serverless computing. These offer greater control but require more maintenance and manual setup to balance workloads. Security demands constant monitoring, and overall costs, including upfront hardware purchases and ongoing maintenance fees, would be higher compared to serverless options.

VI.B.2 Option 2: A Different NoSQL Database Solution

We explored using another NoSQL database for metadata storage. While many NoSQL databases excel in read and write performance and support horizontal scaling, we chose AWS DynamoDB for its exceptional scaling capabilities and seamless integration with AWS security services. Despite varied pricing models among NoSQL databases, DynamoDB's cost-effective pay-as-you-go model perfectly fits our project's budget.

C. *Alternative Processing Solutions*

VI.C.1 Option 1: Traditional Virtual Machines for Media Processing

We explored the use of traditional virtual machines, like EC2 instances, as an alternative to serverless computing for our media processing needs. While EC2 instances offer greater control over performance settings, they demand ongoing administration and maintenance. Scaling EC2 instances can be complex, requiring manual setup and load balancing. Ensuring robust security for these instances also requires continuous oversight.

EC2 instances come with upfront costs for hardware procurement and ongoing maintenance expenses. These combined factors make EC2 instances less cost-effective compared to serverless options.

VI.C.2 Option 2: Investigating Various Serverless Platforms

We also evaluated other serverless solutions besides AWS Lambda. Serverless platforms from different vendors offer varying performance and functionality. However, we selected AWS Lambda due to its seamless integration with other AWS services and strong community support. AWS Lambda provides enhanced security, especially when paired with AWS Identity and Access Management (IAM).

Different serverless platforms have varied pricing models, but AWS Lambda offers precise pricing options, including a free tier for specific usage levels, making it a cost-effective choice for our project.

D. *Alternative Content Delivery Solutions*

VI.D.1 Option 1: Develop CDNs In-House

We considered the ambitious task of creating our own Content Delivery Network (CDN) from scratch. However, replicating the speed and efficiency of a managed CDN like CloudFront would require a substantial investment in equipment and labour. Additionally, self-managing a CDN could introduce complexities and potential issues. The ongoing maintenance costs and initial investment may not be financially sound in the long term.

VI.D.2 Option 2: Use Externally Managed CDNs

We also explored using third-party Content Delivery Networks (CDNs). Well-managed CDNs from reputable providers can deliver excellent performance. We opted for AWS CloudFront due to its seamless integration with other AWS services. AWS CloudFront offers a robust feature set, enabling global expansion by adding new facilities as needed while leveraging the AWS network. Its pay-as-you-go pricing model, combined with certain free usage tiers, made it a cost-effective choice for our project. After careful evaluation, we concluded that AWS services provide an optimal combination of performance, scalability, reliability, security, and cost-effectiveness for our Photo Album application.

VII. CONCLUSION

In conclusion, the proposed solution meets the business requirements of the Photo Album application. It uses AWS services in a way that is highly available, scalable, cost-effective and secure. Use of S3, Cloudfront DynamoDB, Lambda and elastic transcoder all ensure that the website will be fully functional and meet the growing demand of the application. A thorough analysis of alternative options such as on-premises, EC2, and non AWS services has led to the conclusion that the solution proposed in this report is the most suited to the business needs.

REFERENCES

[1] Amazon Web Services, Inc. "Architecture Resources | AWS." Retrieved from https://aws.amazon.com/architecture/.
[2] "AWS-CloudDesignPattern." Retrieved from http://en.clouddesignpattern.org/index.php/Main_Page.
[3] "Architecting for the Cloud AWS Best Practices," 2018. Retrieved from https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf.
[4] "Web Application Hosting in the AWS Cloud," 2019. Retrieved from https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf.
[5] "@IEEEorg, Manuscript Templates for Conference Proceedings," 2020. Retrieved from https://www.ieee.org/conferences/publishing/templates.html.