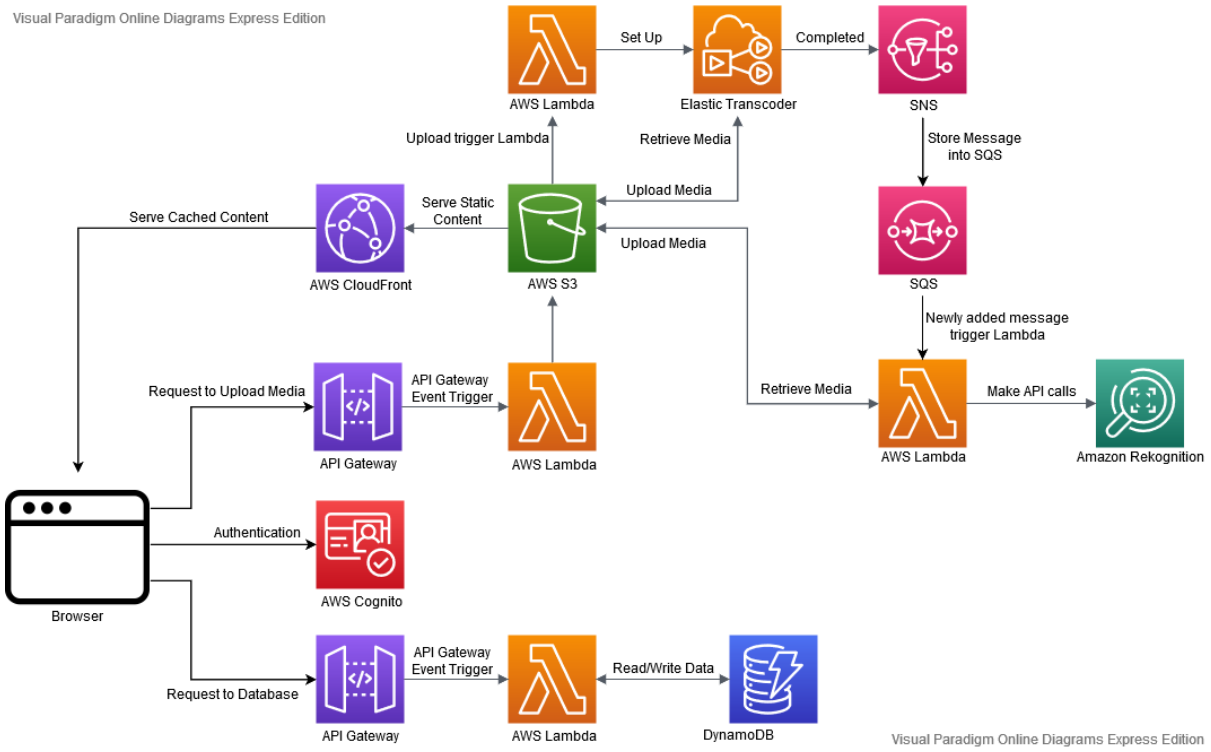
Assignment 3 Lee Zong Yang – 100073484

Web Application Design

Architecture Diagram



AWS S3 will be used to store the photo and media. AWS Lambda could be scaled easily to meet the unpredicted demand in the future. To solve the compute capacity problem face by EC2, the computation task could be breaking down into smaller services and adopt a microservice architecture. This could be easily accomplished by AWS Lambda.

DynamoDB is a NoSQL database managed by AWS and could provide a cost-effective solution while providing great performance in the scenario of a simple database table. To ensure high availability and fast response time to users over the globe, AWS CloudFront which leverage the benefits of edge location is used. From the browser, it uses AWS Cognito for authentication and authorization to make request to upload media or access the database through API Gateway.

Only a few modifications are needed for the Lambda function to set up new task for Elastic Transcoder to handle video media, thus the scalability of the system is great. The Lambda function is set up to trigger upon new media item is uploaded to the S3 bucket and will set up Elastic Transcoder for the media item encoding. Then, the Elastic Transcoder will fetch the media item from S3 and process it.

When Elastic Transcoder has finished the job, it will save the products into S3 and trigger Simple Notification Service with input parameters. After SNS gets triggered, it will and could send out the message to many different parties e.g. SMS, email. In my proposed solution, SNS will send the message to SQS and be stored in SQS queue. Here, another AWS Lambda is set to trigger upon new messages arrive in SQS queue. It could then make API calls to Amazon Rekognition for the image recognition function.

Service Description

The services used in the solution are AWS API Gateway, AWS Lambda, AWS DynamoDB, AWS Cognito, AWS CloudFront, AWS S3, AWS Elastic Transcoder, AWS SNS, AWS SQS, Amazon Rekognition.

AWS Lambda

It is a computing service provided by AWS. The use case of Lambda does not differ much from EC2 but eliminates the need to provision or managing servers. One of the use cases is real-time file processing which is required by the client. The code will be executed by AWS Lambda when a media item had been uploaded to the S3 bucket then set up the Elastic Transcoder to reformat or transcode the media item. The second use case is to copy/read/update/delete data in the

database in response to trigger from AWS API Gateway. The steps to build a functional Lambda function involves write the code to AWS Lambda and set up the event source trigger. Then, the code will only execute when Lambda get triggered while only pay for the computing time used.

AWS API Gateway

It is a fully managed service offer by AWS to simplify the process of creating, publish, maintain, monitor, and secure APIs. It is used in this solution to act as an interface to serverless backend and ease the burden on the application development by providing RESTful APIs to access business logic and data from the backend services. In the case of real-time two-way communication applications, WebSocket APIs is provided too. The processing of a large number of concurrent API calls could be handled easily by API Gateway not to mention authorization, traffic control, CORS support, and API version management.

AWS DynamoDB

It is a key-value and document database (NoSQL) with extremely low latency at any scale. I choice DynamoDB as the persistent layer for my serverless solution because DynamoDB is serverless with built-in high availability and fault tolerance.

AWS Cognito

A SAAS (Software as a Service) provided by AWS for simple and secure user sign-up, sign-in, and access control. AWS Cognito is chosen as an authentication solution because it is a fully managed service without any worries about server infrastructure and reduced cost of implementation as a security specialist is not needed to build own security infrastructure. User authenticated with AWS Cognito will receive a token that could use to authenticate with API Gateway and perform authorize needed action. In this use case, AWS Cognito is crucial to prevent a breach of security.

AWS CloudFront

It is a fast content delivery network (CND) service that stores a cached version of the resource and delivers it to the customers globally with lighting transfer speeds and low latency. In our use case, AWS CloudFront will serve the resources to the customer if it is cached in it. Otherwise, it will fetch the resources from the S3 bucket and cache them for the next request. Moreover, AWS CloudFront also serves the role to protect the content in S3 bucket to expose it to the public.

AWS S3

Amazon S3 is a simple storage service that offers a highly available, durable, and infinitely scalable data storage infrastructure. Through S3, virtually any kind of data in different formats could be stored at any time, from anywhere on the internet. S3 act as a persistent layer to store uploaded and transcoded media item.

AWS Elastic Transcoder

AWS Elastic Transcoder is the alternative solution to transcode the media item without the need of hiring a media item specialist that could cost a lot. By using AWS Elastic Transcoder, the development cycle could be shortened, and the quality of service is promised.

AWS SNS

To promote decoupling in the system and develop serverless applications, AWS SNS is used to provide a publisher/subscriber messaging service for communication between different AWS services. High-throughput, push-based, many-to-many messaging between services can be achieved by publishing/subscribing to corresponding topics. Moreover, SNS could also be used to send out a message to end-users through SMS, mobile notification, and email. SNS is used to notify the media item analysis Lambda function when Elastic Transcoder finishes the job.

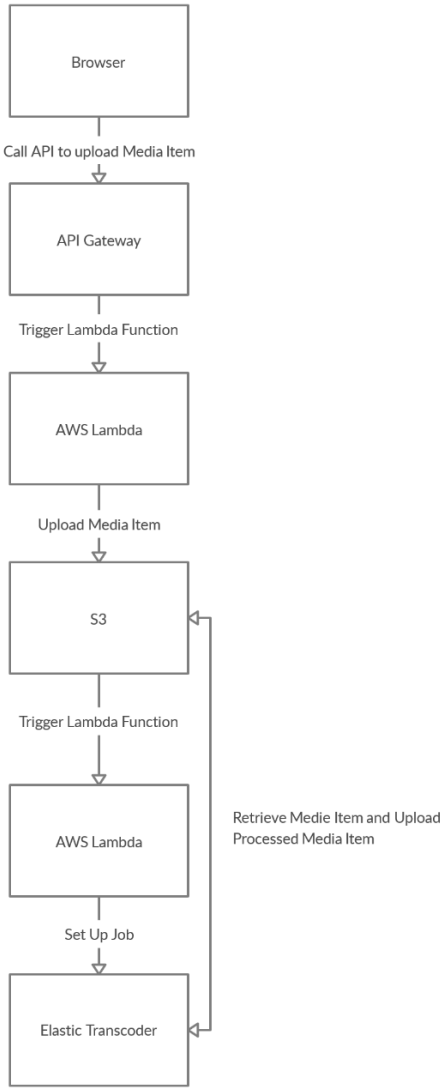
AWS SQS

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

Amazon Rekognition

The analysis of image and video with machine learning could be performed with the use of Amazon Rekognition. The Lambda function will fetch media item from S3 and process them by calling the API of Amazon Rekognition.

Collaboration Diagram



First, the browser will request to upload media item through API Gateway. Then API Gateway will trigger the corresponding Lambda function. The Lambda function will then upload the media item to S3. When new media item is added to S3, respective Lambda function will get triggered and set up Elastic Transcoder for media item handling. Elastic Transcoder will retrieve media item and proceed with its job. After finished the job, Elastic Transcoder will upload the products (different format of the media item) to S3.

Design Rationale

Consistent, secure, and reliable user experience are the most demanding requirements for web applications. To ensure high availability, there is always a need to reserve excessive computing capacity to handle a sudden spike of web requests. Fleets of servers and additional infrastructure required with the traditional server on-premise solution or even with the Amazon Elastic Compute Cloud service, result in extraordinary capital expenditures and extra effort and times for capacity provisioning. However, the troubles of managing servers, prediction of provisioning capacity, or paying for idle resources could be eased by choosing a serverless computing approach that heavily utilizes the Lambda service managed by AWS.

Other than the virtual machine and serverless approaches, there is another solution based on the container. Although docker container needs to run on EC2, AWS provides developers with the capability to configure the container to run on demand. To note that these different approaches do not conflict with each other but could work together to provide a foolproof solution. In this case, the container could be used to lift the execution time limit on Lambda function though orchestrate many Lambda functions together. Provide that the execution time limit for a single invoked Lambda function is 15 minutes, the execution time limit problem rarely exists. Moreover, the computing power for the Lambda function could be increased to shorten the execution time.

In short, with the use of AWS Lambda, the cost could be reduced significantly as there will be no idle time wasted and greatly shorten the time required for development as the developers do not need to manage the OS for the server. Furthermore, the scalability of the system could be improved easily by changing the configuration for computing power as multiple copies of Lambda function could run in parallel. The reliability of the system could be strengthened easily by adopting Micro-Service Architecture using Lambda function. The attribute of Lambda function made it easier to implement Micro-Service Architecture than traditional server solution. This easily triumphs over the virtual machine and container option which need much more configuration.

The choice of database for my solution is DynamoDB which is a NoSQL. Scaling issue is faced by most of the web applications when the user base grows and resulted in increased complexity of data traffic. Thus, the data needs to be processed faster and more robustly. The throughput of SQL is limited to provide great query ability e.g. RDS MySQL has only 1000-80000 IOPS. Thus, standard SQL databases are not always the best fit for a web application. Instead, DynamoDB with the features of high throughput (more than 10 trillion requests per day and more than 20 million requests per second) plus in-memory caching, auto-scaling, backup and restore options could do the job. Judging from AWS previous records, the security should not be a big issue for developers as it is a managed service by AWS thus it would be taken care by AWS and all the connection with the database is encrypted by default.

The primary advantage of DynamoDB is the innate ability to scale horizontally as a NoSQL database and could configure to auto-scale by AWS. This could prevent the problem with performance while decreasing costs. There is also DynamoDB streams that allow other services to receive the changes of data in the database and act on it. Moreover, the need to remove expired data manually by developers is removed as AWS could automatically delete expired items based on timestamps set on the item. DynamoDB is schemaless which is suitable for agile development cycle adopt by most IT teams. By being schemaless, it could offer high-performance queries for data in inconsistent schemas and address query volumes effortless. Furthermore, a snapshot of the data is created regularly for safety purposes.

By using Amazon S3 to hosts web application static assets, the cost could be reduced tremendously as there is no need to pay for idle resources. To provide high availability and global access to the web application, Amazon CloudFront is leveraged to provide a low latency response to the visitors through caching and optimal origin routing, but also reduce the non-necessary workload on the backend. As the static assets in S3 are served through CloudFront, the static assets could be configured to block public access and only allowing authorized services to access it, thus the security is preserved. Amazon Cognito is another way to enforce security for the whole system. Some services could validate the token distributed from Amazon Cognito and perform an action based on the validation result. The security of backend services

could also be enhanced by using API Gateway as an interface to front end application as only the RESTful APIs is visible to the public.

As more features added to the web application, the system's complexities grow by the number of components. Thus, the components need to be decoupled from each other so the reliability of the system could not be compromised as one component outage would affect the whole system. In my solution, SNS and SQS are used to promote decoupling between services with a push paradigm to develop an event-driven paradigm. First, Elastic Transcoder will trigger SNS when it had completed its job. SNS will then put the message into the SQS queue.

Regarding using AWS Lambda together with SQS, there are two different paradigms: push or pull. SQS will trigger Lambda function when there are new messages added into its queue in the case of push. However, AWS Lambda will need to constantly poll the SQS queue in the case of pull, thus lead to higher cost and latency of the system. To note that there is only a pull paradigm for SQS before 2018, thus there are extra efforts needed to use AWS Lambda with SQS. Fortunately, AWS implement SQS as one of the event sources of AWS Lambda in 2018, lead to simplicity in triggering Lambda function from SQS. The benefits of using SNS and SQS together are the messages could be persisted if downstream services are unavailable while immediate reaction could be achieved too and could alleviate pressure on downstream by configuring SQS to throttle queue processing.