DALHOUSIE UNIVERSITY

Master of Applied Computer Science

MACS

CSCI 5409

CLOUD TERM ASSIGNMENT

Report

Ashish Nagpal

Banner ID : B00957622

Email ID : as889330@dal.ca

Table of Contents

Question 1: Introduction

Imagicon is a specialized platform designed to provide users with advanced image sorting and retrieval capabilities. At its core, Imagicon allows users to upload albums and then sort and retrieve specific images based on facial recognition technology. This innovative feature enables users to easily locate and download all pictures containing their face from a particular album. The platform aims to simplify the process of managing and accessing image collections. Whether users are looking to quickly find and download images featuring their face from a specific album or want to organize their image collections more effectively, Imagicon provides a comprehensive solution to meet their needs. Leveraging AWS services, the platform enhances the user experience by automating the image sorting process, making it easier and more efficient for users to manage their image collections.

Target Audience

Imagicon is designed to cater to a diverse audience, including:

- **Individual Users**: Individuals seeking a secure and efficient platform to manage and retrieve their personal photos and images.

- **Photographers:** To share the albums with their clients for the event they have covered.

- **Artists**: Professionals in the art industries looking for a platform to sort, manage, and retrieve specific images from their portfolios.

Question 2: Menu Item Requirements

2.1: Services Selected

Table 1: List of Services Used

Categories	Services Selected for my Application
Compute	**AWS ECS – Fargate with ECR** **AWS Lambda**
Storage	**AWS S3** **AWS DynamoDB**
Network	**Amazon VPC**
General	**AWS SNS,** **AWS Elastic Load Balancer,** **AWS Rekognition**

2.2: Reasons to Select the above-mentioned services other alternatives

1. AWS ECS

Amazon ECS (Elastic Container Service) with Fargate is a fully managed container orchestration service that manage the underlying infrastructure allowing us to focus on containerized applications [1]. I have created Docker images of my frontend (React JS) and backend (Springboot) application and pushed it to ECR. Following this, I have created the ECS cluster and service by defining the task definition which consists of container definition that includes container image and port mapping of container.

Reasons to Choose ECS over other alternatives

- **Scalability**: Imagicon backend service handles various functionalities like uploading images, face recognition using lambda, and SNS notifications. On the frontend side, client makes these request to upload images and search it. Thereby, it was essential to choose a deployment solution that can scale seamlessly based on demand. With containers, it is easy to scale rather than a traditional application, specially, when containers are orchestrated with tools like ECS or Kubernetes. ECS with Fargate provides the ability to scale the application up and down automatically in response to traffic, ensuring that the application remains responsive and efficient without manual intervention.

- **Flexibility**: ECS with Fargate offers a high level of flexibility in terms of deployment configurations. It allows us to define CPU and memory requirements for our containers, ensuring that the application has sufficient resources to operate optimally. For the Imagicon application, Fargate's flexibility was crucial as it allowed us to allocate the necessary resources and manage the containerized application efficiently without the need to manage the underlying infrastructure.

- **Reliability**: The built-in features of ECS, such as automated deployment, health checks, and task placement strategies, ensure high availability and fault tolerance for the Imagicon application. I have configured this using deployment configuration of ECS service which ensures zero down time when a new application version is deployed. Fargate eliminates the need to manage the underlying infrastructure, allowing us to focus on the application logic and ensuring a reliable and robust deployment environment.

- **Security**: ECS offers robust security features that ensure the Imagicon application is protected from unauthorized access and potential security threats. With ECS, we can leverage VPC, security groups, and IAM roles to control access to the application and data, ensuring a secure deployment environment.

- **Cost**: ECS with Fargate follows a pay-as-you-go pricing model, meaning we only pay for the vCPU and memory resources consumed by the containers. This cost-effective pricing model, coupled with the ability to scale resources based on demand, ensures that the Imagicon application is deployed in a cost-efficient manner, optimizing the operational costs.

2. **AWS Lambda**

In my Imagicon application, I have the features to add a photo album and search the images in which they are present. Whenever a user adds the images, the backend server API is triggered which uploads the images to S3 and invokes the Lambda function. The lambda function calls the rekognition API and creates the collection using the photo album.

Reasons to Choose AWS Lambda over Alternatives

- **Scalability**: Lambda function has the ability to scale automatically based on request made by the user.

- **Reliability**: Lambda automatically runs code in multiple Availability Zones to ensure high availability and fault tolerance. However, in case of an EC2 instance with EBS volume manual intervention is required to recover from failures, leading to potential downtime and operational challenges.

- **Customizability with Logging**: It is easy to write code for lambda function without any prior installation. The code can also be updated through console which makes it fast for testing. AWS Lambda integrates with Amazon CloudWatch Logs to provide real-time monitoring and logging to our infrastructure. It also provides insights into performance, errors, and resource utilization, and enabling efficient troubleshooting and optimization.

- **Cost Saving**: The biggest advantage of using Lambda is the pay-per-use pricing model. We are only charged for the compute time which is consumed i.e the execution time of lambda, with no additional charges when the code is not running. Thereby, in my application as I am only executing the rekognition service API when the user calls the API, thus continuous running of server in not required. Moreover, Lambda automatically scales and allocates resources, optimizing resource utilization and reducing costs by eliminating the need to manage idle resources.

- **Serverless Architecture**: AWS Lambda provides seamless integration with ECS and DynamoDB that enables a serverless architecture. This allows you to focus on building and deploying applications without managing the underlying infrastructure and ensures efficient data processing and management

3. **S3**

In my application Imagicon, I have used AWS S3 to store images of user which they want to share with their friends and the image which is used for searching the face. The images stored in AWS S3 bucket – “imagicon-bucket-uploads” is used for further processing by rekognition service.

Reasons to Choose AWS S3 over other alternatives

AWS S3 is an object storage that allows us to store and retrieve any amount of data, including images, videos, and backups, making it highly flexible.
Although, Amazon S3 has a flat structure instead of a hierarchy structure as in a file system, we can still store files in folders by using shared name prefix for the grouped objects [2]. In my application, I am storing images for users in different folders and for search images, there is a separate folder.

Mostly, users are concerned about the data loss, especially their images which has a connection. Since, Amazon S3 is designed for 99.999999999% (11 9's) of durability, ensuring that the data is highly available and durable against failures, errors, and data loss with built-in replication and versioning users can safely store images without any fear.
I have also provided public readOnly access for images using Access Control List so that the entire bucket is not exposed publicly that ensures security. Also, S3 offers data encryption at rest and in transit to protect the stored data from unauthorized access and data breaches.

Other alternatives like AWS DynamoDB, AWS Neptune, AWS Aurora, and AWS Athena are mostly suited for handling structured data due to their powerful query capabilities.

4. AWS DynamoDB

I have a DynamoDB table for the user details which stores the FriendsList and rekognitionCollectionId. Username is taken as the partition key. The FriendsList is a list of ARNs of SNS topics which the user has created.

Reasons to choose AWS DynamoDB over another alternatives

- **Scalability**: DynamoDB, you can easily increase or decrease read and write capacity based on the workload, ensuring consistent performance and availability as the application grows.

- **Reliability**: The built-in replication and multi-AZ deployments of Amazon DynamoDB ensure high availability and fault tolerance. DynamoDB automatically replicates data across multiple AWS regions to provide data durability and availability, even in the event of regional failures or outages.

- **Flexibility**: DynamoDB is flexible in storing and querying NoSQL data. I am using DynamoDB with Java SDK that has DocumentClient, which simplify the process of interacting with the database and integrating it with the application code.

5. Network Amazon VPC

Amazon Virtual Private Cloud (VPC) is a web service offered by Amazon Web Services (AWS) that allows customers to provision a logically isolated section of the AWS Cloud with advanced security features such as security groups and network access control lists (NACLs) to enable inbound and outbound filtering at the instance and subnet level.

VPC service offers high availability and fault tolerance by allowing customers to distribute their instances and resources across multiple Availability Zones within a region. This ensures that if a single Availability Zone becomes unavailable, customers can still maintain high availability and continuity of their applications and services.

Reasons to Choose Amazon CloudFront over other alternatives

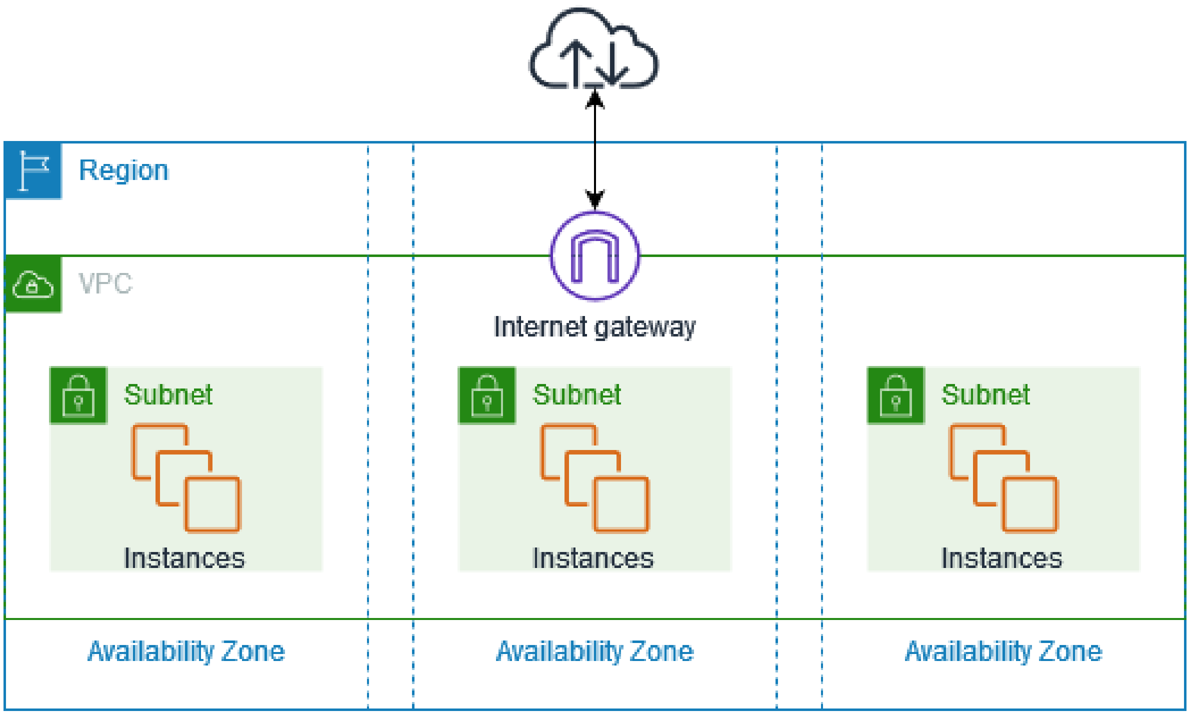

Figure 1: Vpc Diagram [9]

Why I use Amazon VPC

I am using Amazon VPC to have an isolated network so that the services are not exposed directly to the world. This will help in secure environment and less networking threats.

Reasons to Choose AWS VPC over other alternatives

Amazon Virtual Private Cloud (VPC) offers seamless integration with a variety of AWS services, providing a robust and flexible networking environment. Within VPC, I have created ACLS, security groups route table and public and private subnets using internet and NAT gateway to optimize the performance and availability of the application.

In my infrastructure, I have configured the ECS and Lambda services within a private subnet of the VPC, ensuring a secure and isolated environment for running the application workloads. The Elastic Load Balancer (ELB) is deployed in a public subnet of the VPC to handle incoming client requests and forward them to the ECS service.

Amazon VPC offers robust security features and capabilities to ensure the confidentiality, integrity, and availability. I have implemented security groups and network access control lists (NACLs) to control the inbound and outbound traffic to the ECS service, restricting access only from the Elastic Load Balancer, and enhancing the overall security posture of the application.

6. AWS SNS

Reasons to Choose AWS SNS over other alternatives

For my application Imagicon, I choose AWS SNS for my Friends List feature in which a user can create multiple friends list. A friend list is basically a SNS topic created in the AWS. Along with friend list the user also has to add the email addresses of the friends to send the notification. The email addresses acts as a subscription for that particular friend list (SNS Topic).

Once the user has created their photo album, they can share the link to their friends and the friends who have subscribed the SNS Topic will receive the email.

Simple Notification Service (SNS) works on Fan-out model that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

SNS was the most suitable service in my case because I wanted to send broadcast email to each subscriber (Friend) present in the SNS Topic. (Friend List).

Also considering the future scope of the application, numerous communications protocols are supported by SNS, including HTTP, HTTPS, SMS, email, and push notifications for mobile devices. As a result,

integrating with various endpoints and clients would be easy. I can extend the feature of to have in-app notification for users.

It is easy to use SNS with SDK as well. With just a simple method definition, a message can be published to all the subscribers.

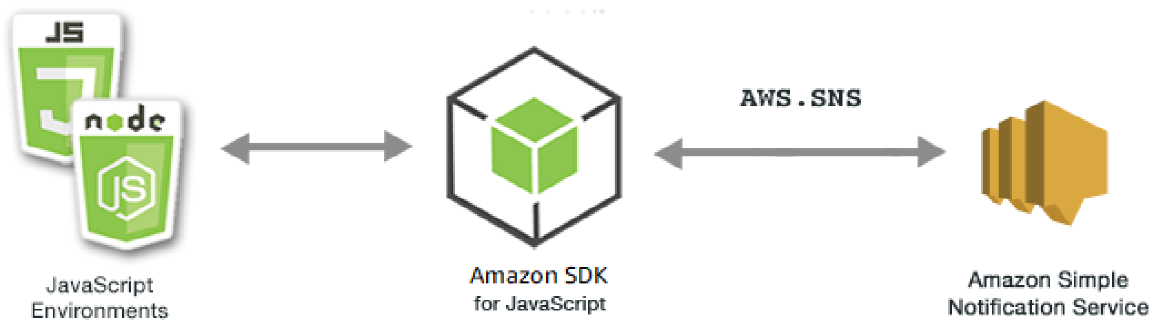

Figure 2: SNS using SDK [10]

7. Amazon Elastic Load Balancer

Why I choose Amazon Elastic Load Balancer

Amazon Elastic Load Balancing (ELB) is a service provided by Amazon Web Services (AWS) that automatically distributes incoming application traffic across multiple targets in multiple Availability Zones. ELB acts as a single point of contact for clients, balancing the load and ensuring high availability and fault tolerance of applications.

Reasons to choose Elastic Load Balancer

Being in the public subnet, the Elastic Load Balancer (ELB) plays a vital role in the network architecture of Imagicon application. It efficiently manages and distributes incoming client requests to the ECS service located in the private subnet. This configuration not only safeguards the frontend and backend servers, but also optimizes the performance by efficiently distributing the incoming traffic to the ECS Fargate.

ELB provides essential features like health checks and automatic scaling, which enhance the reliability and scalability of the Imagicon application. The security group associated with the ECS service is configured to only allow traffic from the Load Balancer, which ensures that only validated and load-balanced requests are forwarded to the application. This setup effectively protects against potential DDoS attacks and other malicious activities by acting as a barrier between the public internet and the private ECS resources.

8. Amazon Rekognition

Why I choose Amazon Rekognition

Amazon Rekognition, provided by Amazon Web Services (AWS), is a deep learning-based image and video analysis service that can identify objects, people, text, scenes, and activities, as well as detect any inappropriate content. The service's accuracy and reliability ensure that the facial recognition and image searching functionalities are performed with high precision, contributing to the overall performance and effectiveness of the application.

The best part of using Amazon Rekognition is that it involves direct usage of APIs without the prior knowledge of machine learning. Instead of storing the images in collections directly, it basically stores a mathematical representation of the face detected in the images called the face vectors. The face vectors are used to match and search the faces, which makes the service fast.

Reasons to choose Amazon Rekognition

Amazon Rekognition is the core service for Imagicon Application that uses its image processing capabilities. The ImageController in the Imagicon backend is responsible for handling interaction with Rekognition via Lambda function which is acting as a microservice. When a user uploads images via the /uploadBulk API, the create_collection lambda function is invoked which creates the collection inside Rekognition service and add faces to it by processing the images from the S3 bucket. Furthermore, the /searchImage API utilizes Rekognition to compare the uploaded image with the existing collection, returning the matched faces and their corresponding S3 path stored in ExternalImageId of the matched face.

Due to its powerful visual analysis even without requiring any machine learning expertise, I chose Amazon Rekognition.

Question 3: Deployment Model

For my Imagicon website, the Public Cloud Deployment Model is the most suitable choice. The following are the reasons why I chose the Public Cloud Deployment Model as the most suitable one:

- **Scalability and Cost Efficiency:**

My Imagicon website is hosted on various AWS services, a public cloud provider. This model allows me to leverage the scalability of AWS services, such as Lambda, ECS, DynamoDB, and S3, to meet the demand for my application. In addition, I only pay for the resources I use, which can lead to cost savings compared to maintaining a private cloud.

- **Global Accessibility:**

Since Imagicon is a globally accessible website, it needs to be available to users worldwide. Public cloud providers like AWS have a global presence, which ensures low latency access for users across the globe, providing a seamless user experience.

- **No Need for Private Infrastructure:**

  Imagicon does not handle highly sensitive data (like financial or health records) that might require the extra security controls of a private cloud. While user authentication is necessary (and handled by AWS Cognito), a public cloud deployment provides sufficient security measures for a website like Imagicon.

- **Less Maintenance Effort:**

  With a public cloud model, AWS is responsible for the maintenance, updates, and infrastructure security, reducing the operational burden on my team and allowing us to focus more on developing and improving the Imagicon website.

Let’s briefly discuss why the other models would be less suitable:

- **Private Cloud:**

  This model is generally chosen when an organization needs to handle sensitive data and requires more control over its environment. It’s typically more expensive and requires more maintenance effort than a public cloud. Given the nature of Imagicon, a private cloud does not seem necessary.

- **Community Cloud:**

  This model is used when multiple organizations with similar requirements and goals share the infrastructure. It doesn’t seem to apply to the Imagicon website.

- **Hybrid Cloud:**

  This model is beneficial when some data or applications are kept on a private cloud or on-premises due to regulatory requirements while other services are deployed on a public cloud. Imagicon does not seem to have such requirements.

- **Multi-cloud:**

  This model involves using multiple cloud services from different providers. While it can offer benefits in terms of avoiding vendor lock-in and increased reliability, it also adds complexity in terms of managing and integrating services across different platforms. Given the current architecture of the Imagicon website, which exclusively uses AWS services, a multi-cloud model doesn’t seem necessary.

So, considering the nature and requirements of the Imagicon website, a Public Cloud Deployment Model on AWS is the most suitable choice.

Question 4: Delivery Model

4.1 Describe your delivery model. Why did you choose this delivery model?

The Software as a Service (SaaS) Delivery Model is the best-suited choice for my web application. Below are the reasons why I chose the SaaS Delivery Model:

Overview of the SaaS Delivery Model:

SaaS is a software licensing and delivery model in which software is accessed online via a subscription, rather than being bought and installed on individual computers. In this model, the software vendor hosts and maintains the servers, databases, and the code constituting the application.

Advantages of SaaS for Imagicon:

- **Accessibility:**
  As a SaaS application, Imagicon can be accessed from any device with an internet connection. This means that users can upload, search, and manage their images from anywhere, on any device, providing them with a seamless and convenient experience.

- **Maintenance and Upgrades:**
  In the SaaS model, the service provider that is AWS is responsible for maintaining the application and performing updates. This means that users don't have to worry about maintenance, and they always have access to the latest version of the application with new features and improvements.

- **Cost Effectiveness:**
  With SaaS, users don't need to invest in hardware to host the application, which reduces their upfront costs. Additionally, the costs for the user can be much lower than traditional software licensing since the cost of software, hardware, and staff are spread out over a larger number of users.

Why SaaS is Chosen Over Other Delivery Models:

- **Platform as a Service (PaaS):**
  PaaS would involve providing a platform allowing customers to develop, run, and manage applications without the complexity of infrastructure maintenance. Imagicon is already developed and doesn't require users to modify or develop it further, making SaaS a more appropriate model.

- **Infrastructure as a Service (IaaS):**
  IaaS provides users with the most flexibility but also requires them to manage more of the system. Users would have to handle everything from the application down to the virtualization layer, which isn't necessary or appropriate for Imagicon.

- **Public Cloud:**
  Given the global accessibility requirement and the scalability of Imagicon, a public cloud deployment model complements the SaaS delivery model perfectly. The Public Cloud provides the necessary infrastructure and services that support the SaaS delivery model effectively.

So, considering the nature and requirements of the Imagicon website, a Software as a Service (SaaS) Delivery Model hosted on a Public Cloud is the most suitable choice.

Question 5: Final Architecture

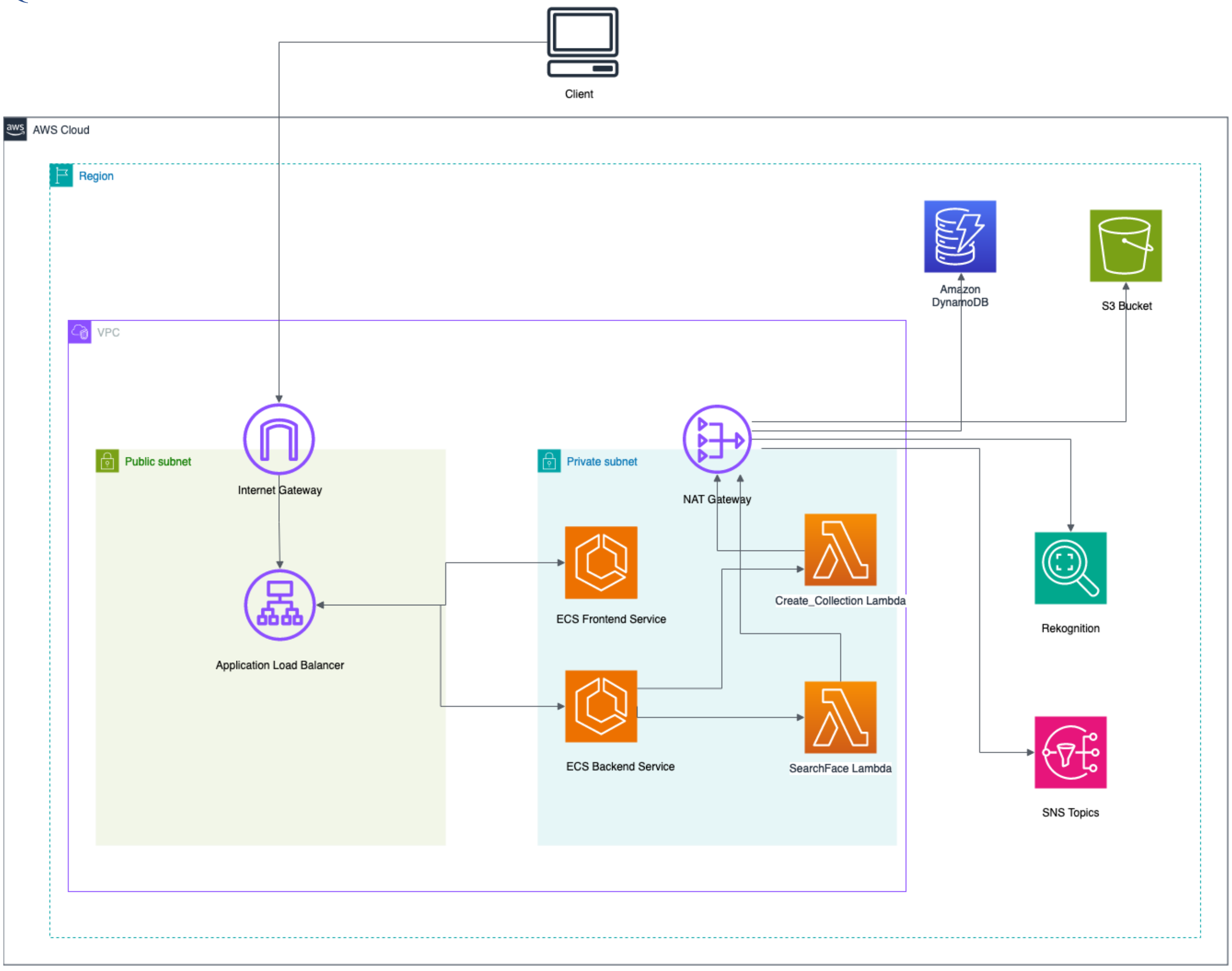

Figure 3: Infrastructure Imagicon [11]

5.1: How do all of the cloud mechanisms fit together to deliver your application?

The architecture of my cloud-based application "Imagicon" is designed to efficiently handle image uploads, processing, and searches.

Frontend and Load Balancing: The user interacts with the frontend of the application, which is hosted on ECS. The requests from the frontend are then routed through an Elastic Load Balancer (ELB) to ensure even distribution across backend resources and to enhance the application's availability and fault tolerance.

Backend and Data Processing: The backend is powered by a Spring Boot application deployed on ECS, utilizing AWS Fargate for container orchestration. AWS Fargate eliminates the need to manage servers, providing a serverless compute engine for running containers. This ensures scalability without the overhead of managing infrastructure.

Storage and Database: Imagicon leverages Amazon S3 to store the uploaded images securely. DynamoDB is used as the database to store user information and to keep track of the image collections created by the users. The Lambda functions are utilized to process the images, recognize faces, and perform searches.

Integration and Communication: The Spring Boot application communicates with various AWS services using the Java SDK client. The Lambda functions are invoked to interact with the Amazon Rekognition API for image processing and face recognition using python's Boto3 client. When a user uploads an image, the Lambda function reads the image from the S3 bucket, performs face recognition using Rekognition, and updates the DynamoDB table with the recognized faces and corresponding image URLs.

Security and Access Control: All the resources are placed within a VPC to provide an additional layer of security. The security groups associated with the ECS service and the Lambda functions are configured to allow traffic only from the Elastic Load Balancer. The ECS and Lambda functions access the DynamoDB and S3 resources through the NAT gateway so that there is only outgoing traffic.

Following is the workflow of my Application:

When a user uploads an image through the **/uploadBulk** API, the Spring Boot application uploads the image to the S3 bucket and invokes the create_collection Lambda function to create a Rekognition collection and add faces to it. For the **/searchImage** API, the Spring Boot application retrieves the image from S3 and calls the searchFaces Lambda function. The Lambda function uses Rekognition to search for faces in the image and returns the matched faces' S3 paths stored in ExternalImageId [3] key of the face.
The FriendListController manages the SNS topics and subscriptions. When a user wants to add friends or resend a confirmation, it adds subscriptions to the existing SNS topic. Messages are published to the SNS topics through the **/publishMessage** API, and Lambda functions are triggered to send messages to the subscribed users. The ViewImageController checks the existence of a Rekognition collection by querying the DynamoDB table.

Why I have chosen this Architecture?

This architecture provides a scalable, reliable, and cost-effective solution for the Imagicon application. The combination of ECS, Lambda, S3, DynamoDB, and VPC ensures seamless integration, efficient image processing, and secure data storage. The use of serverless computing with Lambda functions reduces the operational overhead. The security measures implemented using VPC and security groups ensure that the application is secure and compliant with best practices.

5.2: Where is data stored?

For the Imagicon website, the data is stored in various AWS services to ensure efficiency, scalability, and reliability.

- **Amazon S3 (Simple Storage Service):**
All uploaded images are stored in Amazon S3, a scalable object storage service offered by AWS. S3 provides a secure and durable storage solution, and it is used to store the images uploaded by users.

- **DynamoDB:**
User information and the collectionId of user are stored in Amazon DynamoDB, a NoSQL database service provided by AWS. DynamoDB offers seamless scalability and low-latency performance, making it ideal for storing and retrieving user data and face collections efficiently.

- **Amazon Rekognition:**
The detected face data and facial recognition results are stored in the Amazon Rekognition service itself, which is a deep learning-based image and video analysis service. Rekognition provides comprehensive and accurate face detection and recognition capabilities.

5.3: What programming languages did you use (and why) and what parts of your application required code?

For my Imagicon website, the architecture consists of a client-side application and a server-side application. The frontend of the Imagicon website is built using ReactJS, a JavaScript-based framework known for its efficiency in creating interactive and dynamic user interfaces. ReactJS was chosen for its ability to handle the image upload feature and provide a seamless user experience.
The backend of the Imagicon website is developed using Java (Spring Boot) for the business logic and Python (Lambda Functions) for specific tasks like image processing and face recognition using the Amazon Rekognition API.
Lambda is invoked by the Springboot backend server to perform specific task. It is created for image processing and face recognition using the Amazon Rekognition API. I have written the lambda code in python language as it is easy to debug code and comprehensive support of Boto3 client for interacting with various AWS services.
The communication between the frontend and the backend is established using axios, a popular JavaScript library used to make REST API calls. The backend is hosted on ECS. The choice of Java for the backend was based on its robustness, scalability, and extensive community support provided by the Spring Boot framework. Spring Boot simplifies the process of building production-ready applications by providing default configurations and eliminating much of the boilerplate code.

5.4 How is your system deployed to the cloud?

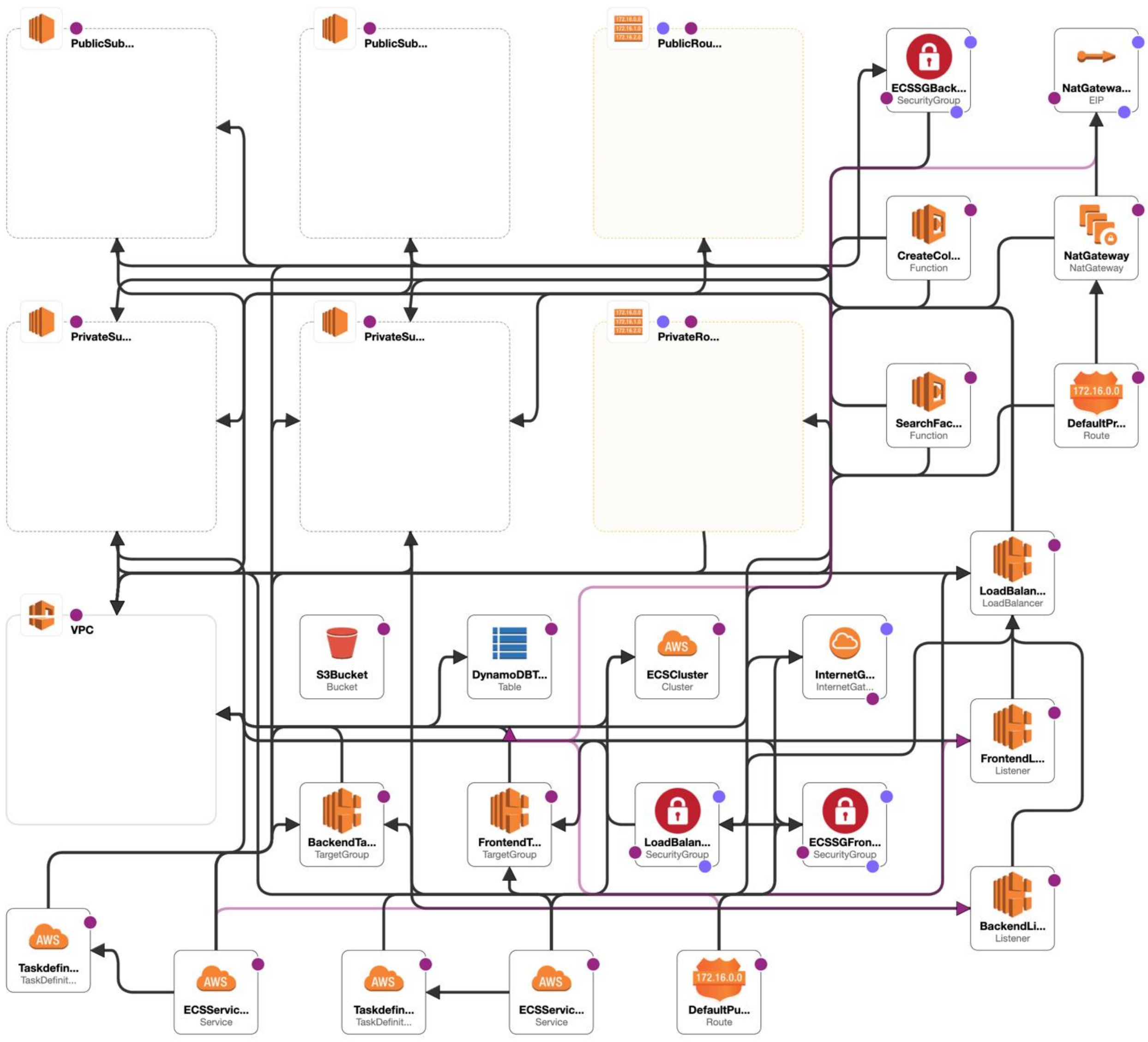

Figure 4: CloudFormation Architecture

For the deployment of the Imagicon website to the cloud, I have adopted an automated and scalable approach using AWS CloudFormation along with other AWS services.

I have utilized AWS CloudFormation to provision and manage the infrastructure resources required for the Imagicon website. With CloudFormation, I can describe all the AWS resources needed for the application in a template file. This template file is then used to create and provision the resources in an automated and repeatable manner, ensuring consistency and reducing the manual effort in deploying the application.
The Imagicon application is containerized using Docker. AWS ECS (Elastic Container Service) is used to manage and orchestrate these Docker containers. The containers are deployed as tasks within ECS clusters, which are provisioned and managed by CloudFormation. The serverless component of the application is deployed as AWS Lambda functions. To ensure security and isolation, the application resources are deployed within a Virtual Private Cloud (VPC). CloudFormation provisions the necessary VPC, subnets, and security groups to ensure the application's network resources are securely deployed and accessible. An Elastic Load Balancer (ELB) is used to distribute incoming application traffic

across multiple targets, such as ECS tasks. The ELB configuration is also defined in the CloudFormation template to ensure the proper distribution of incoming traffic.

5.5 If your system does not match an architecture taught in the course, you will describe why that is, and whether your choices are wise or potentially flawed.

The Imagicon application uses the Service-Oriented Architecture (SOA) approach to a certain extent to deliver its functionality efficiently and effectively. It separates the application into distinct services that can operate independently, communicate with each other via well-defined interfaces, and provide specific functionalities, which are the core principles of SOA.

1. **Microservices with AWS Lambda and ECS:**

   The Imagicon application employs a microservices architecture utilizing AWS Lambda for specific functionalities like image processing and face recognition, and AWS ECS (Elastic Container Service) for managing containerized services.

2. **Infrastructure as Code with CloudFormation:**

   For infrastructure provisioning and management, Imagicon leverages AWS CloudFormation, a method consistent with SOA principles to ensure repeatable and consistent service deployments.

3. **Data Management with Amazon S3 and DynamoDB:**

   Imagicon uses Amazon S3 for image storage, DynamoDB for NoSQL database needs. This aligns with the SOA approach of selecting the right data storage services tailored to meet specific service requirements.

Question 6: Data Security at different layers

6.1 How does your application architecture keep data secure at all layers? Which security mechanisms are used to achieve it

1. **Network Level Security**
   The Imagicon application is deployed within a Virtual Private Cloud (VPC) to create a logically isolated section in the AWS Cloud, which helps to control network traffic and enhance the security. Security groups and network ACLs are implemented to manage inbound and outbound traffic to the application, ensuring that only necessary and authorized communication is allowed. Utilizing a VPC, security groups, and network ACLs adds an extra layer of security to the application by controlling the traffic flow and minimizing potential security risks and unauthorized access.

2. **Compute Level Security**
   ECS tasks and Lambda functions are hosted within private subnets to prevent direct internet access, enhancing the security posture of the application. IAM roles and

policies are utilized to control and manage access to the compute resources, ensuring that only authorized entities can interact with the application components. Running compute resources within private subnets and utilizing IAM roles and policies helps to protect the Imagicon application from potential threats, vulnerabilities, and unauthorized access, thereby maintaining the confidentiality, integrity, and availability of the resources.

3. **Data Storage Security**
   Data at rest and in transit within DynamoDB is encrypted using AWS Key Management Service (KMS), providing an additional layer of security and safeguarding the data from potential breaches and unauthorized access. Access Control List is enabled for objects to be available publicly instead of Bucket policies. Implementing encryption for data at rest and maintaining multiple versions of objects in S3 ensures the security, integrity, and availability of the data, protecting against potential data loss, breaches, and unauthorized access.

4. **Application Layer Security**

   Rekognition is used for image and video analysis, providing secure and accurate identification of objects, scenes, and activities, enhancing the security and reliability of the application. SNS provides monitoring capabilities through Dead Letter Queues (DLQ), facilitating the detection and handling of email delivery failures and other potential incidents. The Elastic Load Balancer (ELB) serves as a critical component that directs incoming web traffic across multiple ECS instances, acting as a layer between the user and the actual server infrastructure.

6.2: Identifying Vulnerabilities and Addressing Them:

While the Imagicon website employs a robust security approach, there are potential vulnerabilities that need to be addressed to enhance the overall security posture of the application. By implementing the proposed solutions and adhering to security best practices, we can mitigate these vulnerabilities, ensuring the confidentiality, integrity, and availability of data and safeguarding against potential security threats and incidents

1. **Inadequate IAM Role and Policy Configuration**
   Currently in the academic environment we have limited access to modify the IAM roles and policies which may result in unauthorized access to the AWS services and resources, leading to potential data breaches and security vulnerabilities.
   To prevent that we can review and audit IAM roles and also Implement Multi-Factor Authentication (MFA) for added security when accessing the AWS Management Console.

2. **Insufficient Monitoring and Logging**
   The services lack proper monitoring and logging due to which it may be challenging to detect and respond to security incidents timely which can potentially compromise the integrity and availability of the application.

To solve this we can enhance monitoring capabilities by integrating AWS GuardDuty to gain insights into user activities and detect any suspicious activities or potential security breaches. Regularly review and analyze access logs and set up alerts for any unusual or unauthorized access attempts.

3. **Data Protection:**
   Without encryption for data in transit, sensitive data may be exposed to interception and potential data breaches during transfers.
   We can implement AWS Certificate Manager (ACM) to provision and manage SSL/TLS certificates, enabling encryption for data in transit, safeguarding the confidentiality and integrity of the data.

4. **Application Layer Security:**
   There is a risk of users uploading inappropriate or malicious images, which may compromise the security and integrity of the application and its resources. Implement content validation and filtering mechanisms to detect and prevent users from uploading inappropriate or malicious images. We can utilize Amazon Rekognition to analyze and verify the content and authenticity of the images uploaded by users to prevent the injection of attacks through images and ensure the security and integrity of the application.

Question 7: Estimated cost to reproduce this architecture in private cloud

Following is a breakdown provides an overview of the specific hardware and software components required and their estimated costs for reproducing the architecture of the Imagicon website in a private cloud environment.

1. **Hardware Infrastructure:**
   - **Dedicated Servers**: Powerful servers capable of handling the application's workload, particularly for image recognition and processing.

     Hardware Required:

     CPU: AMD EPYC 7F32 (8 cores) [5]

     RAM: 64GB DDR4

     Storage: 1TB NVMe SSD

     Estimated Cost Calculation:

     $5,604 per server x 5 servers = $28,020

   - **Networking Equipment:**

     Routers, Switches, and Firewalls:

     Essential for secure network traffic management and to protect against unauthorized access.

Hardware Required:

Cisco Catalyst 9300 Series Switches

Palo Alto Networks PA-220 Firewalls

Estimated Cost Calculation:

$3,500

2. **Virtualization and Orchestration Software:**

   Virtualization software to create virtual machines (VMs) on the physical servers, maximizing server utilization.

   Software Required:

   VMware vSphere Enterprise licenses [6] for 5 servers

   Estimated Cost Calculation:

   $4,780 per server x 5 servers = $23,900

3. **Networking and Security Software:**
   - **Firewall and Security Software**:

     Palo Alto Networks Firewall: Robust security measures, including firewalls and security software, to protect the private cloud environment.

     Software Required:

     Palo Alto Networks PA-Series Firewall software for 5 servers

     Estimated Cost Calculation:

     $4,000 - $6,000

   - **Load Balancing Software:**

     Nginx or HAProxy: Load balancing software to distribute incoming traffic across multiple servers for optimal performance and high availability.

     Software Required:

     Nginx [7] or HAProxy setup for load balancing

     Estimated Cost Calculation:

     $2,200

4. **Database Software:**
   For the Imagicon website, a reliable database management system is essential to store and manage the application's data effectively. In line with our current setup utilizing

Amazon DynamoDB, the database service will be pivotal in ensuring both data integrity and accessibility.
The estimated cost, including data replication for fault tolerance, is approximately $2,500 per server. Given that we are planning to deploy on 5 servers, a minimum of 5 storage setups would be required. Therefore, the total estimated cost for the database setup would be $12,500.

Question 8: Which cloud mechanism would be most important for you to add monitoring

For the Imagicon website, the most important cloud mechanism to implement effective monitoring would be Amazon CloudWatch integrated with AWS Rekognition. By leveraging the combined capabilities of AWS Rekognition and Amazon CloudWatch, we can enhance the monitoring and security of the Imagicon website, ensuring the quality and appropriateness of the images being uploaded, and protecting the platform from potential security threats and vulnerabilities associated with image uploads [4].

Amazon CloudWatch is a comprehensive monitoring and observability service provided by AWS that allows for real-time monitoring of various AWS resources, applications, and services. Integrating CloudWatch with AWS Rekognition will enable continuous monitoring and analysis of the image content being uploaded by users, ensuring the quality, appropriateness, and security of the images on the Imagicon platform.

- **Real-time Monitoring with CloudWatch**: By integrating AWS Rekognition with Amazon CloudWatch, we can set up real-time monitoring and alerts for the image content uploaded by users. CloudWatch can monitor the Rekognition service's API calls, providing immediate notifications for any inappropriate or malicious content detected in the uploaded images, enabling timely actions to address potential security threats and vulnerabilities.

- **Image Content Moderation with Rekognition**: AWS Rekognition can analyze images uploaded by users in real-time to identify and flag inappropriate or explicit content. By leveraging CloudWatch's monitoring capabilities, we can continuously monitor the Rekognition analysis results, ensuring that only suitable and safe images are displayed and shared on the Imagicon platform, thereby preventing the upload of censored or wrong images and maintaining the platform's credibility and user trust.

- **Protection Against Injection Attacks**: Integrating AWS Rekognition with Amazon CloudWatch allows for continuous monitoring of the image content and analysis results. This enables us to implement additional security measures to detect and prevent image-based injection attacks, such as injecting malicious code or scripts into images. CloudWatch can trigger alerts for any suspicious or potentially harmful image content detected by Rekognition, safeguarding the platform from potential security threats and vulnerabilities.

Question 9: Future Aspects of My Application

There are several features and enhancements which can be considered to improve the functionality, performance, and user experience of the application.

1. **Content Delivery Network (CDN) for Image Optimization:**

   Improve the loading speed and performance of the website by optimizing the images and delivering them efficiently to the users. We can utilize AWS CloudFront, a global content delivery network (CDN) service, to deliver the images efficiently to the users from the nearest edge location, reducing the latency and improving the website's performance.

2. **User Authentication and Authorization:**

   We can implement a robust user authentication and authorization mechanism to enhance the security and protection of the user data and resources. We can utilize AWS Cognito, a robust user identity and authentication service, to implement secure and scalable user authentication and authorization mechanisms to protect the user data and resources effectively.

3. **Automated Image Tagging and Categorization**

   Enhance the user experience by implementing automated image tagging and categorization features. By utilizing AWS Rekognition's image recognition capabilities, the application can automatically detect and tag objects, scenes, and activities within the images uploaded by users, allowing for better search functionality and organization of images. We can use AWS Rekognition for image analysis and tagging, Amazon DynamoDB for storing image metadata and tags.

4. **Enhanced Security with Data Encryption and Compliance:**

   We can implement enhanced security measures by encrypting sensitive data and ensuring compliance with data protection regulations. By utilizing AWS Key Management Service (KMS) and AWS Identity and Access Management (IAM), the application can encrypt sensitive data at rest and in transit, and enforce strict access controls to protect user data and ensure compliance with data protection regulations. AWS KMS can be used for data encryption and AWS IAM can b used for access control and compliance management

References

[1] “Amazon ECS on AWS Fargate,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS_Fargate.html. [Accessed: April 09, 2024].

[2] “Organizing objects in the Amazon S3 console by using folders,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-folders.html [Accessed: April 09, 2024].

[3] “IndexFaces API Reference, “*Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/APIReference/API_IndexFaces.html#rekognition-IndexFaces-request-ExternalImageId.[Accessed: April 09, 2024]

[4] “Monitoring Rekognition with Amazon CloudWatch,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/rekognition-monitoring.html.[Accessed: April 09, 2024]

[5] “Rax qh12-12e4,” *Thinkmate.com*. [Online]. Available: https://www.thinkmate.com/system/rax-qh12-12e4. [Accessed: April 09, 2024].

[6] “VMware vSphere pricing in 2024: Licensing and overhead costs,” *V2 Cloud*, 28-Mar-2023. [Online]. Available: https://v2cloud.com/blog/vmware-vsphere-licensing-and-costs. [Accessed: April 09, 2024].

[7] *G2*. [Online]. Available: https://www.g2.com/compare/f5-nginx-vs-haproxy. [Accessed: April 09, 2024].

[8] T. Robinson, “Private cloud pricing,” *OpenMetal IaaS*, 17-Nov-2023. [Online]. Available: https://openmetal.io/cloud-deployment-calculator/. [Accessed: April 09, 2024]

[9] “What is Amazon VPC?, ” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. [April 09, 2024].

[10] “Amazon Simple Notification Service Examples,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/sns-examples.html. [April 09, 2024].

[11] “Flowchart maker & online diagram software,” *Diagrams.net*. [Online]. Available: https://app.diagrams.net/. [Accessed: April 08, 2024].