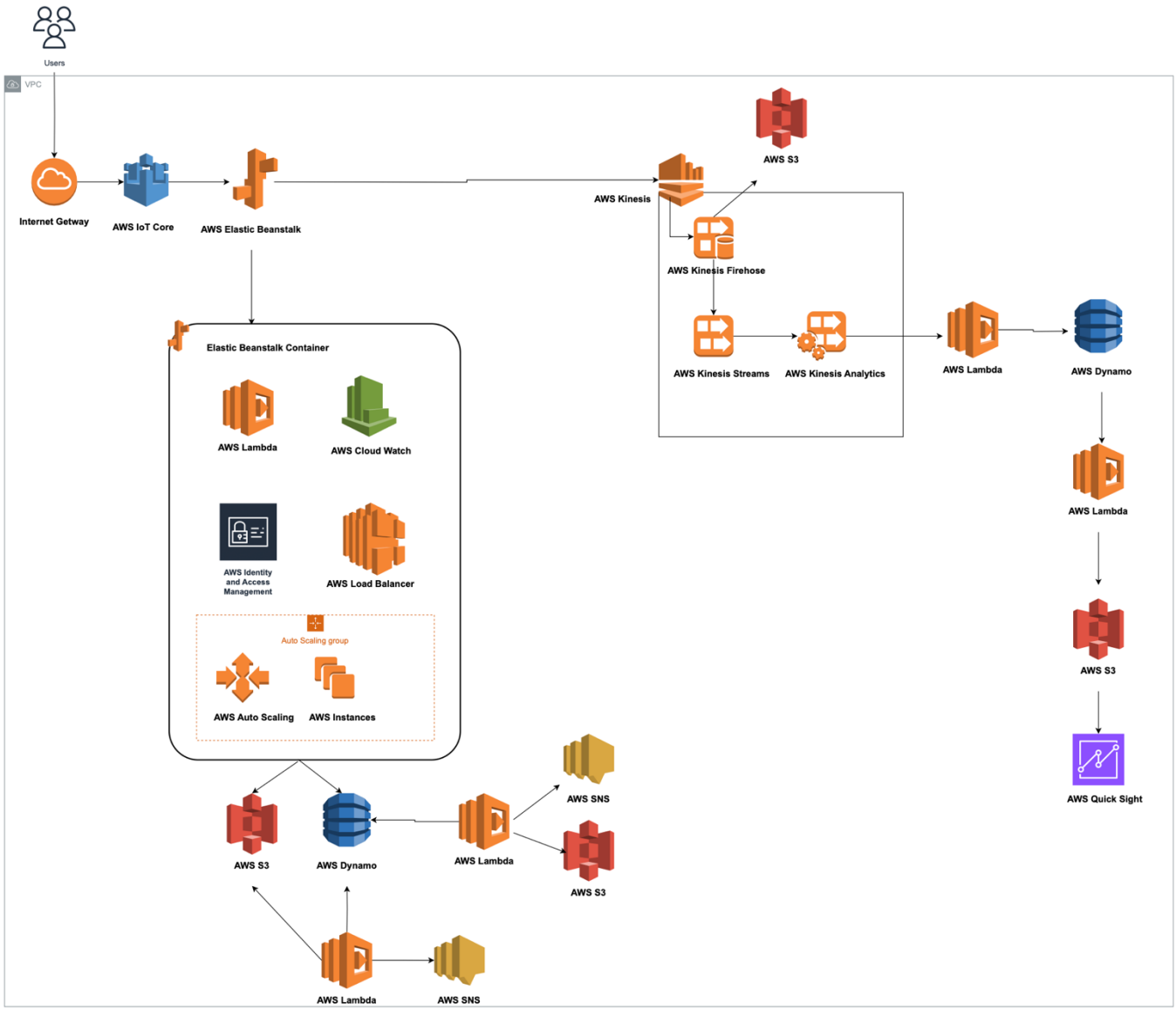
FIT5225 2024 S1
Assignment 2

Designing an AWS Cloud-Based Solution for a Parking Management Application

Name: Peichun Shih
Student ID: 33475881
Tutors: Jay Zhao, Qifan Deng, Jinchun Du

Table of Content

I. Architectural Diagram



II. Design Explanation

The application is designed to put on the virtual private cloud, and it is based on AWS services. AWS Internet Gateway service is used to allow communication between instances in a virtual private cloud and the internet.

1. Entry Registration

As a vehicle arrives at the parking area, the camera captures the image of the license plate, and the image data will be sent to AWS IoT Core for processing. To manage the parking application, AWS Elastic Beanstalk service, which is a Platform as a Service (PaaS), is adopted. Within the service, there is an Elastic Beanstalk Container that includes built-in capabilities, such as load balancing and auto-scaling to ensure performance (Scott & Guillois, n.d.). In addition, Elastic Beanstalk service integrates with AWS CloudWatch service to monitor the application health and performance metrics. Furthermore, it incorporates AWS Identity and Access Management service to control access to AWS services and resources with security measures. IAM facilitates the management of users, groups, and roles, as well as provides security to the cloud infrastructure (What Is IAM? - AWS Identity and Access Management, n.d.). When the camaras send data to IoT Core, the execution of AWS Lambda functions will be triggered. The functions will process the incoming images, extract, and store license plate information, such as license plate number, entry timestamp, any other related data, in AWS DynamoDB. Moreover, the captured vehicle image will be stored in AWS S3 for further reference and backup.

2. Parking Duration Monitoring

For monitoring parking durations, AWS Lambda service will be configured to run continuously, making sure that parking duration information in DynamoDB is timely updated. These functions will be periodically check and update the duration time for each parked vehicle.

3. Overstay Detection

For the overstay detection process, AWS Lambda functions are used to monitor parking durations actively. When detecting a vehicle surpasses the allowed free parking duration, the Lambda functions will mark the vehicle as a violator. To ensure transparency and timely communication, the system will notify the vehicle owners with AWS SNS service, which helps to send real-time messages or notifications to multiple platforms or individual recipients, which includes email, SMS, HTTP, and mobile push notifications (*Push Notification Service - Amazon Simple Notification Service - AWS*, n.d.). It assists with immediate notifications to

vehicle owners concerning the exceeded time and prompts them to take necessary actions. Subsequently, the functions will proceed to record the corresponding parking fee and update the existing data in DynamoDB. The update includes details on exceeded parking duration, status, as well as extra fee incurred. The continuous monitoring mechanism not only helps the effective implementation of the parking regulations but contributes to efficient parking management.

4. **Plate Verification**

   AWS Lambda functions will be triggered to process and extract the image data captured at the exit point. Throughout the process, the license plate number will be extracted and queries within DynamoDB to find out the corresponding entry records, and further compare the entry image stored in AWS S3 for the double confirmation of the identity of the vehicle. Then, the system leverages the extracted information to verify the duration of stay and other related data within DynamoDB. By cross-referencing data from DynamoDB and S3, the system builds a reliable process to validate parking activities.

5. **Exit Process and Ticket Issuance**

   AWS S3 will be used to store both vehicle images and the ticket template. As the vehicle prepared to exit the parking area, AWS Lambda functions will be triggered to not only capture the exit image but also retrieve relevant data from DynamoDB and S3. The data retrieval leads to the generation of electronic tickets with essential details, such as the license plate number, entry and exit times, as well as the parking fee. The system will leverage AWS SNS service to send the electronic ticket to the vehicle owner's mobile app. After the payment is received, the Lambda function will manage the system to open the gate, while simultaneously logging the payment details in DynamoDB and archiving the exit images in S3 respectively.

6. **Reporting and Analytics**

   In this design, the vehicle parking data is regarded as the streaming data. Consequently, AWS Kinesis service is utilised to manage real-time data streams. AWS Kinesis Data Firehose allows for easily capturing and ingesting streaming data. It is used to simplify the process of managing streaming data (*Amazon Firehose - Streaming Data Pipeline - Amazon Web Services*, n.d.).

   Additionally, AWS Kinesis Data Streams service helps to build custom applications for processing streaming data. With this service, the system is able to control over essential aspects, such as data partitioning, data retention, and data processing logic. It enhances the system's capability to flexibly adapt to varying

data requirements and processing demands (*Real-Time Streaming Analytics - Amazon Kinesis Data Streams - AWS*, n.d.).

Moreover, AWS Kinesis Analytics service is the tool adopted to analyse streaming data in real-time. By continuously evaluating and analysing incoming data streams, Kinesis Analytics ensures that the analysed results are always current that reflect the latest data trends and insights. Subsequently, when the real-time analysis is completed, the Lambda function is triggered to transforms the results to the specific format suitable for storage within AWS DynamoDB.

Then, the Lambda function is programmed to convert the necessary data stored within DynamoDB into a format compatible with the requirements of AWS QuickSight service, such as CSV or Parquet. AWS QuickSight is a BI service, which creates interactive dashboards and displaying data visualisations. The transformed data is archived within AWS S3 and QuickSight service will prepare for the data visualisation (*What Is Amazon QuickSight? - Amazon QuickSight*, n.d.). The data visualisations thus generate the insights derived from the parking data and can be used to improve future parking management.

To address key aspects, the service strategies are used and explained as follows:

- **Scalability**

  AWS Elastic Beanstalk is employed to manage the application since it provides built-in capabilities for auto-scaling and load balancing. It is proficient in dynamically dealing with fluctuating levels of traffic, which allows for optimal performance despite varying loads. In addition, the design incorporates AWS Kinesis Data Firehose and Kinesis Data Streams to ingest real-time streaming data. These services provide the flexibility to adapt to shifting data loads and data partitioning. Accordingly, the scalability of the system is enhanced to manage and process the streaming data from the parking management application.

- **Security**

  In this design, AWS Identity and Access Management is applied to manage access to AWS services and resources. It regulates access privileges by providing different permissions for users, groups, and roles within the system. This approach guarantees a secure environment for the data and the infrastructure.

- **Failure Handling**

  To improve the fault tolerance and reliability of the application, AWS CloudWatch is utilised to vigilantly monitor the health of the application,

enhancing the proactive detection and swift response to potential failures. Additionally, AWS Lambda functions are designed to incorporate error handling and retry mechanisms to cope with failures, ensuring uninterrupted operation across diverse operational scenarios. By integrating these services, the system not only improves fault tolerance but also build a resilient operational framework which can navigate unexpected challenges and maintaining optimal performances.

- **Cost Effectiveness**

  In this architechtural design, AWS Lambda is integrated for serverless computing, while AWS S3 is applied for scalable storage solutions. By leveraging these AWS services with pay-as-you-go pricing models and highly cost-effective pricing options, it allows the system to optimise resource utilization and expenditure. The strategy not only facilitates efficient resource allocation but also maintains the system's flexibility and scalability to accommodate diverse requirements and dynamic workloads.

In conclusion, the proposed AWS architectural design provides a comprehensive and efficient approach for monitoring and regulating parking activities. It offers a reliable and scalable platform to efficiently manage parking operations, enhance security measures, handle failures effectively, optimise operational costs. The design lays a solid foundation for elevating overall parking management experiences for both operators and users.

References

Amazon Firehose - Streaming Data Pipeline - Amazon Web Services. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/firehose/

Push Notification Service - Amazon Simple Notification Service - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/sns/

Real-Time Streaming Analytics - Amazon Kinesis Data Streams - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/kinesis/data-streams/

Scott, T., & Guillois, J.-B. (n.d.). *Deploy containers by using Elastic Beanstalk - AWS Prescriptive Guidance*. AWS. https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/deploy-containers-by-using-elastic-beanstalk.html

What is Amazon QuickSight? - Amazon QuickSight. (n.d.). https://docs.aws.amazon.com/quicksight/latest/user/welcome.html

What is IAM? - AWS Identity and Access Management. (n.d.). https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html