MONASH University

FIT5225 Assignment2 AWS Cloud Solution Assessment

Student Name: Ziqi Pei, zpei0003@student.monash.edu

2024 S1

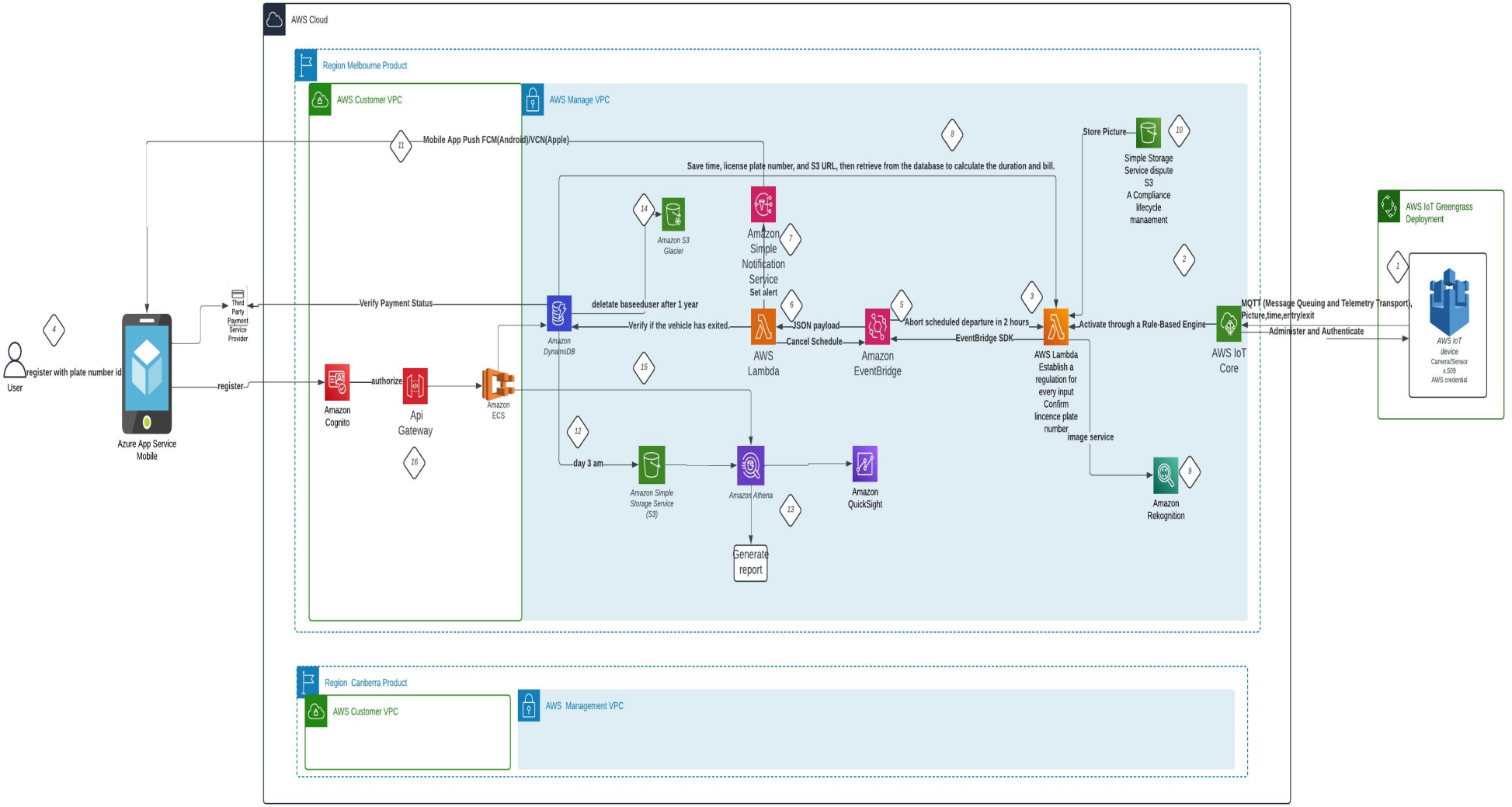
Tutor:Jay Zhao, Jinchun Du(Goldi),Qifan Deng

Solution Overview and Background

The parking management system uses license plate recognition technology with cameras at the entrance and exit. It automatically records vehicles entering and exiting, tracking their stay duration. Billing reminders are sent via a mobile app, and electronic tickets are issued for overtime parking. Real-time alerts notify managers of such events for prompt action. Data analysis tools help managers optimize operations, including space utilization.

1.Architecture Design

I used the official AWS Architecture Icons to draw the solution architecture diagram clearly showing the key components of the system and their interaction relationships.


AWS Cloud
Region Melbourne Product
AWS Customer VPC
AWS Manage VPC
Region Canberra Product
AWS Customer VPC
AWS Management VPC
AWS IoT Greengrass Deployment
User
register with plate number id
register
Azure App Service Mobile
Third Party Payment Service Provider
Verify Payment Status
Amazon Cognito
authorize
Api Gateway
Amazon ECS
Amazon DynamoDB
deletate baseeduser after 1 year
Verify if the vehicle has exited.
Amazon S3 Glacier
Amazon Simple Notification Service
Set alert
AWS Lambda
JSON payload
Cancel Schedule
Amazon EventBridge
Abort scheduled departure in 2 hours
EventBridge SDK
AWS Lambda Establish a regulation for every input Confirm lincence plate number
Activate through a Rule-Based Engine
Store Picture
Simple Storage Service dispute S3 A Compliance lifecycle manaement
AWS IoT Core
MQTT (Message Queuing and Telemetry Transport), Picture,time,entry/exit
Administer and Authenticate
AWS IoT device Camera/Sensor x.509 AWS credential
Mobile App Push FCM(Android)/VCN(Apple)
Save time, license plate number, and S3 URL, then retrieve from the database to calculate the duration and bill.
image service
Amazon Rekognition
day 3 am
Amazon Simple Storage Service (S3)
Amazon Athena
Amazon QuickSight
Generate report
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

2. Architecture Description

1. IoT Device Access: Use AWS IoT Core service to connect and manage a large number of IoT devices (such as cameras, vehicle detectors, etc. at the entrance and exit of the parking lot). IoT Core can assign X.509 certificates to devices to ensure device identity is trusted and communication is secure.

2. Data Reporting: IoT devices report data using the MQTT protocol through the message broker provided by IoT Core. Messages will carry information such as license plate images and timestamps, and be securely transmitted to the cloud through the WebSocket Secure (WSS) protocol.

3.Trigger Image Processing: IoT Core can directly trigger Lambda functions through the Rules Engine. After receiving a new image, it will immediately invoke Lambda for license plate recognition and verify whether the recognition result matches the registered license plate number.

4.User Registration: Parking lot users can register accounts through a mobile app or webpage, submitting information such as license plate numbers. The registration request will reach the backend service through Amazon API Gateway, and the relevant user metadata will be stored in Amazon DynamoDB.

5.Scheduled Task Creation: When a vehicle enters, the Lambda function will call the Put Events API of EventBridge to publish a scheduled reminder event. EventBridge Rules can flexibly set the target and execution time of the event, for example, sending a message to a certain Lambda function after 2 hours.

6.When a vehicle exits, the system repeats the license plate recognition process and updates information such as exit time to DynamoDB. After the EventBridge scheduled event is triggered, it will check whether the corresponding license plate has exited. Those still in the parking lot are considered overtime and need to send a reminder to the user.

7. When sending reminders, the Lambda function will publish push messages to the corresponding topic through the Amazon SNS service. SNS supports various information publishing protocols, such as mobile app notifications, SMS, email, etc. For users who have bound the app, the system uses Firebase Cloud Messaging (FCM) and Apple Push Notification service (APNs) to send notifications.

8. The license plate recognition results will be written to the DynamoDB table, together with the vehicle entry time, as the basis for parking timing. DynamoDB is a fully managed NoSQL database that can provide stable single-digit millisecond performance and automatically scale to meet high-concurrency read and write demands.

9. Image Recognition: The system calls Amazon Rekognition for text extraction (OCR) and recognition of license plate images. Compared to building a model from scratch, Rekognition is a mature managed image recognition service. Using it can significantly reduce system complexity and automatically scale to handle request peaks.

10. Cold Data Storage: The uploaded license plate images will be directly saved to the S3 bucket. With the help of S3's version control and lifecycle management functions, it can meet the needs of data retention and backup. At the same time, S3's high availability and elastic scalability can cope with growing data storage needs.

11. Payment Service: If the vehicle is still in the parking lot, it indicates overtime. At this time, SNS will push an alert message to the vehicle owner's mobile app. For Android and iOS devices, SNS uses Firebase Cloud Messaging (FCM) and Apple Push Notification service (APNs) respectively to implement remote notifications.

12. Data Archiving: In order to optimize cost and performance, the system will migrate historical data that is not frequently accessed in the DynamoDB database to Amazon S3 at 3 am every day. This process is automatically completed by AWS Data Pipeline Migration Service.

13. Data Analysis: The parking lot operation team can use Amazon Athena to directly query historical data on S3 to achieve ad-hoc queries and analysis. By visualizing the analysis results through Amazon QuickSight, managers can gain insights into key indicators such as parking space utilization and optimize the allocation of parking lot resources.

14. Data Archiving: Considering compliance and cost, the system has set a data retention policy. The DynamoDB table only retains data for the most recent year, and data from before will be automatically archived to S3 Glacier.

15. Data Backup: Amazon DynamoDB provides automatic, asynchronous table backup functions. The system will regularly create full backups of the table to ensure data can be recovered.

16. Users can register license plates and contact information through the app or web page. The system uses Amazon Cognito to manage user identity pools and implements functions such as registration and login. API Gateway exposes secure HTTPS interfaces, and Lambda functions can read and write user data in DynamoDB after identity verification.

3.1 Scalability

To ensure that the system can handle growing traffic and data volumes, I adopted several AWS services that can automatically scale in the design:

(1) .Use AWS IoT Core to connect and manage a large number of IoT devices (cameras and vehicle detectors). It can handle requests from billions of devices and scale within milliseconds.

(2). The license plate recognition program is deployed on Amazon ECS, using Fargate launch type and Auto Scaling, which can automatically adjust the number of ECS tasks (containers) based on the number of tasks and resource utilization.

(3). Use Amazon S3 to store license plate images. S3 can handle thousands of requests per second with almost unlimited capacity.

(4). Use AWS Lambda to interact with other AWS services and perform backend logic such as billing and notifications. Lambda can scale to thousands of concurrent requests within milliseconds.

3.2 Security

As a charging system, security and compliance are critical. I have taken the following key measures:

(1). All sensitive data (user identity information, license plate numbers, billing records) are encrypted at rest and in transit, using AWS KMS to manage keys uniformly.

(2). All data access must go through identity authentication and authorization, and no data can be accessed without authorization.

(3). CloudTrail records all API calls throughout the process to ensure that operations are auditable and traceable.

(4). S3 storage buckets prohibit anonymous public read and write, and have enabled server-side encryption.

(5). IoT device certificates are uniformly managed by IoT Core and can be revoked or rotated at any time. Communication between devices and from devices to the cloud is encrypted.

(6). Services within VPC communicate through private network segments and are isolated from the public network.

3.3 Fault Handling

The system follows the recommendations of the AWS Well-Architected Framework and implements high availability design at all levels:

(1). Compute Layer: Lambda and Fargate are regionally deployed and can be scheduled across availability zones. Even if one availability zone is unavailable, it will not affect the service.

(2). Storage Layer: S3 and DynamoDB use a deployment across 3 availability zones by default, with high durability.

(3). Multi-Region: The system deploys two environments in Australia, and can be switched to the disaster recovery region when the main region is unavailable.

(4). Monitoring: CloudWatch monitors system metrics in all aspects, and notifies operation and maintenance personnel through SNS when abnormalities occur.

(5). Backup: In addition to cross-AZ replication of storage itself, data is also backed up to S3 and replicated across regions.

3.4 Cost Optimization

Using the Serverless service family and usage-based pricing model, the system can minimize costs:

(1). Lambda and Fargate are pay-as-you-go, charged entirely based on the number of invocations or resource usage, with no waste.

(2). DynamoDB and Aurora Serverless can also automatically scale in and out, reducing the resource cost during idle times to a minimum.

(3). S3's tiered storage automatically converts infrequently accessed data to lower-cost storage classes.

(4). The system divides multiple functions into independent Lambda functions, making deployment more flexible, cold starts faster, and also facilitating fine-grained cost control.

(5). Athena's serverless data analysis makes analysis costs strictly linearly related to data volume, without incurring additional management overhead.

Compared with traditional self-built parking management systems, using AWS cloud-native Serverless architecture not only greatly improves functionality, performance and security, but also significantly reduces the total cost of ownership

(TCO) of the system. A well-designed Serverless architecture may cost less than $1 per parking space per month.

Note: The above report refers to authoritative resources such as AWS official blog, App backend:https://aws.amazon.com/appsync/ and certification website during the architecture design process, and uses tools such as Aws calculator: https://calculator.aws/ to assist in the design.

The content does not use Chatgpt