Cloud Computing for Connected Autonomous Vehicles

Paritosh Yogesh Shelke, 110059982,

Mohammad Zuhaib Mansoor, 110062386

Abstract

Cloud technologies are rapidly emerging across the world. They have transformed various industries and businesses. From transportation to healthcare and hospitality. Taking businesses to the cloud has proven efficient in terms of reducing costs and improving overall performance, resulting in increased profits. The solution designed takes these factors into the account for creating an underlying framework for autonomous vehicles which eradicates dependency on physical hardware for storing the data generated by various sensors on board. This framework is completely hosted on the AWS cloud, utilizing a wide range of services provided by Amazon. These services are namely S3 used for storage of raw data, AWS Lambda for creating functions for various services ranging from vehicle registration to anomaly detection, generating driver safety score, Dynamo DB which is a No-SQL database that stores the processed data coming from Amazon Kinesis services that act as a data-stream. Finally, Amazon SNS generates over-the-air updates for the user. The data stored in the Dynamo DB can be utilized for the learning of machine learning and deep learning models thus, extending this framework in making an autonomous system. Another use case would be to create a data lake that stores this enormous data, which can be accessed across the globe by software developers, contributing to the autonomous vehicle system [2].

I. INTRODUCTION

Connected autonomous vehicles are shaping the future of the automotive industry. With onboard sensors and cameras installed, an autonomous vehicle can drive and navigate from source to destination with little to no human intervention. These vehicles are often designed to communicate with each other and thus are connected. There are various types of sensors like LiDAR, radar, cameras, and GPS assisting these vehicles for localization, positioning, and navigation purposes. The data being generated from these sensors is enormous and requires a reliable and scalable service to host which can be processed further. The cloud plays a vital role in facilitating this flow. With no on-premises hardware required, higher computational power, and uninterrupted networking capabilities, it is possible to enhance the performance of connected autonomous vehicles, focusing on the development of AV software. The services utilized in designing this framework are namely S3 or Simple Storage Service used for storing raw data. This is an object-based storage service that stores data in the form of objects. AWS Lambda is used for creating functions for various services ranging from vehicle registration to anomaly detection and generating driver safety scores. This is a service that provides serverless and scaling capabilities. Furthermore, Dynamo DB is a No-SQL database that stores the processed data coming from Amazon Kinesis service that acts as a data stream. Finally, Amazon SNS generates over-the-air updates for the user. The data stored in the Dynamo DB can be utilized for the learning of machine learning and deep learning models.

II. IMPLEMENTATION DIAGRAM

To better understand the flow of processes through the AWS services used in this system, an implementation diagram of the system is designed (Figure **1**). The implementation of this system is based on the data from physically tested sensors in the vehicles, however for this project, to avoid hardware dependency, a simulation is being used. The sensors of the vehicle are simulated in real-time, and the data gets collected and then gets fed to the system in the diagram below. The first AWS service that can be seen in the diagram is the IoT Core. It is dependent on the hardware sensors; however, it is still included in this diagram as it plays a crucial role in the hardware implementation of this solution. There are a few use cases that are highlighted in this diagram that can be divided so that each process is clearly understood.

The Just-in-time Registration (JITR) use case simplifies the process of registering a new vehicle with the AWS IoT. The JITR function triggers AWS Lambda that activates a signed, unknown vehicle certificate, and attaches a custom security policy to it, enabling it to be used for authentication and authorization with AWS IoT [1].

The second use case that can be seen in the diagram is Anomaly Detection. It uses Amazon Kinesis as the main service which consists of Data Firehose, Data Analytics and Data Streams. This use case detects outliers in the streaming data that is sent to an amazon S3 bucket as raw sensor data through Data firehose delivery system. Then the Kinesis Data Analytics service analyses each record, extracts anomalies, and sends those records to a Kinesis Stream, triggering an AWS Lambda Function that stores the data into an AWS DynamoDB table and notifies the driver of any issues using the AWS SNS service [1].

The trip data is another important use case that is used in this system. The vehicle sensor data that is previously processed in the vehicle is sent to this system at regular intervals and then stored in an Amazon DynamoDB table. Upon completion of a trip, a driver safety score is generated by a microservice that uses AWS Lambda, which uses the aggregated trip data based on the speed, idling, acceleration, braking, cornering etc. And then the driver is notified of the score subsequently. This concept is used in a lot of different areas such as insurance companies use this concept to provide discounted prices to the customers who achieve a good score at the end of the year [1].

The final use case is the diagnostic trouble code (DTC) detection use case. It is a simple but valuable service that improves vehicle ownership and maintenance experience. When DTCs are detected by AWS IoT, it invokes a Lambda function that stores the DTC in an Amazon DynamoDB table, translates the DTC into layman's term, and then alerts the driver of any issues using the SNS service [1].

This Connected Vehicle Solution uses AWS services that are completely serverless and manageable along with providing scalable, modular, API-driven applications. Therefore, the user does not need to worry about patching, managing, or scaling the infrastructure to meet the requirements for more users or vehicles. Also, it leverages a pay-as-you-go price model so that there is no need to worry about software licenses, long term subscription or contracts. The user pays only for whatever service they use. As a result, the users can focus on creating and designing innovative applications and solutions instead of worrying about overhead management and administrative tasks [1].

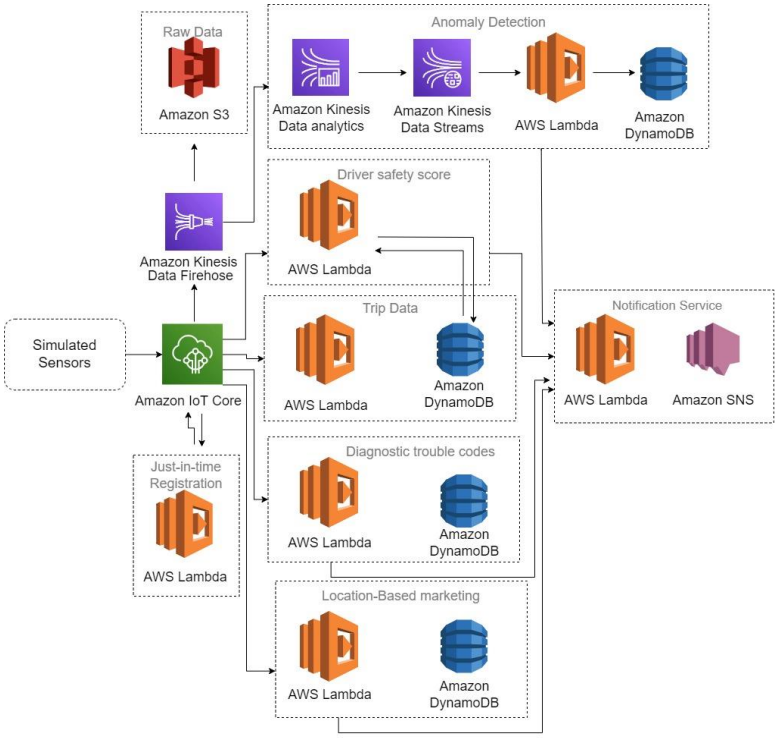

Figure 1: Implementation Diagram

III. RESULTS

A. *Simulation Results*

The images shown below are the results of the simulations and the working of the system that is designed for this project. Figure 2 and Figure 3 are the screenshots from the simulator that is used to simulate the sensors from a vehicle. The simulator allows us to register a vehicle and create virtual sensors that can be used to get data accordingly. This data shows the timestamp when it was recorded, a virtual Vehicle Identification Number (VIN) of the vehicle, torque, engine speed, fuel level, oil temperature and the location of the vehicles. In Figure 4, a map shows a pinpoint at a location in Toronto, Ontario. This pinpoint shows where the virtual vehicle has stopped after driving along the pathways for a user-specified time; in this case it was 30 seconds.

Figure 2: Simulation Result 1

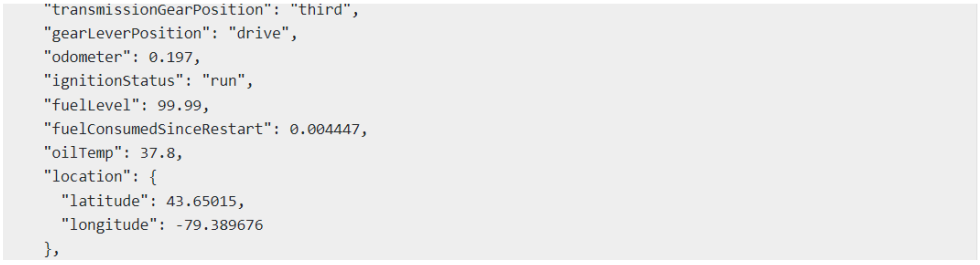

Figure 3: Simulation Result 2

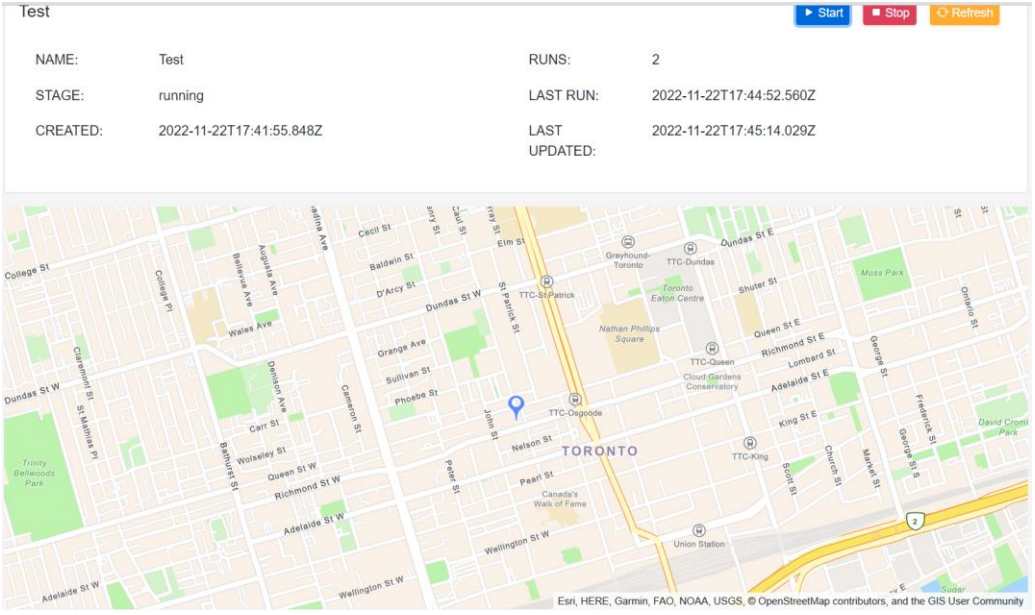

Figure 4: Simulation Result 3

B. Tabular Results

The data generated from the sensors of the simulated vehicles is stored in the DynamoDB tables from where the data can be transferred to the main system and processed further. The content in the DynamoDB is stored in the form of JSON files. Figure 5 shows the first table JSON file attributes which contains the time when a vehicle is registered and other details of the vehicle. The other details of the simulations are stored on another table. The JSON file attributes of the simulation data are shown in Figure 6 which contains details of the simulation such as status, timestamps, duration of the simulation, intervals etc.

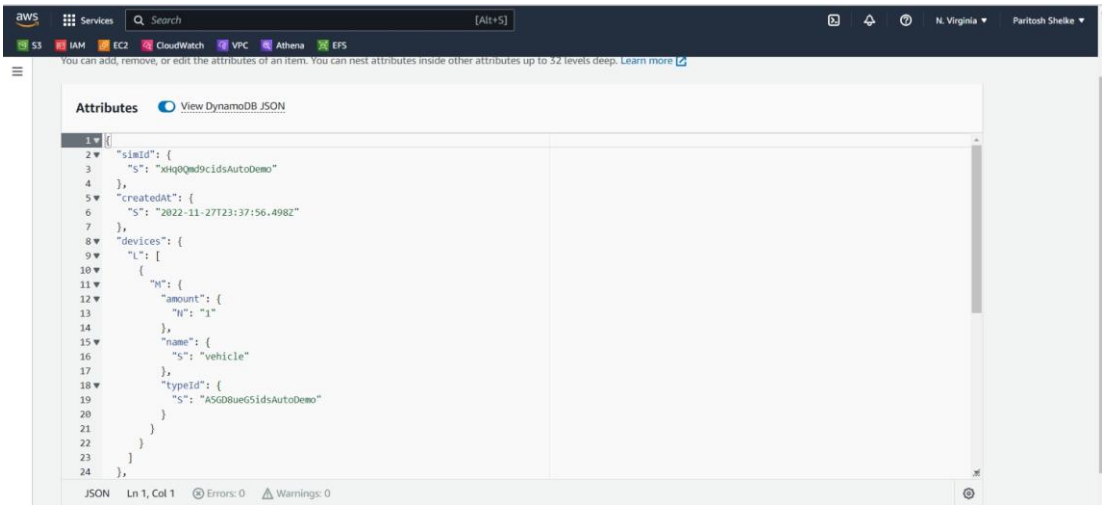

Figure 5: Table results 1

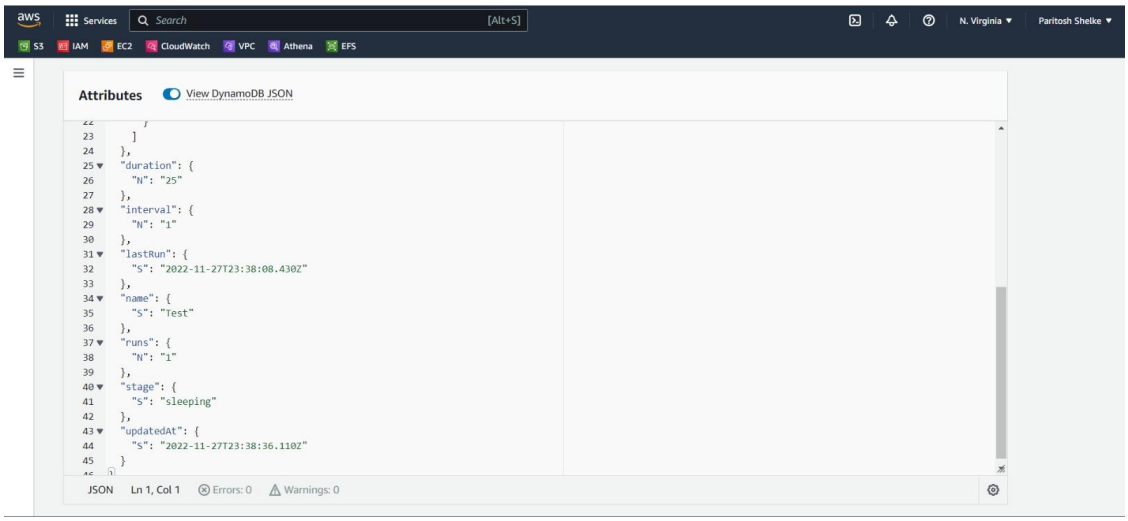

Figure 6: Table results 2

The tables can be exported from DynamoDB to other services. This is shown in **Figure 7** and Figure 8 where DeviceTypesTable is the table that contains vehicle data and SimulationsTable is the table that contains Simulation Data. The tables can be exported directly to an S3 bucket to process further, or Amazon Kinesis can be used to deliver the data as streams to the buckets for further processing. Kinesis can be more effective when the data is continuously recorded in the tables.

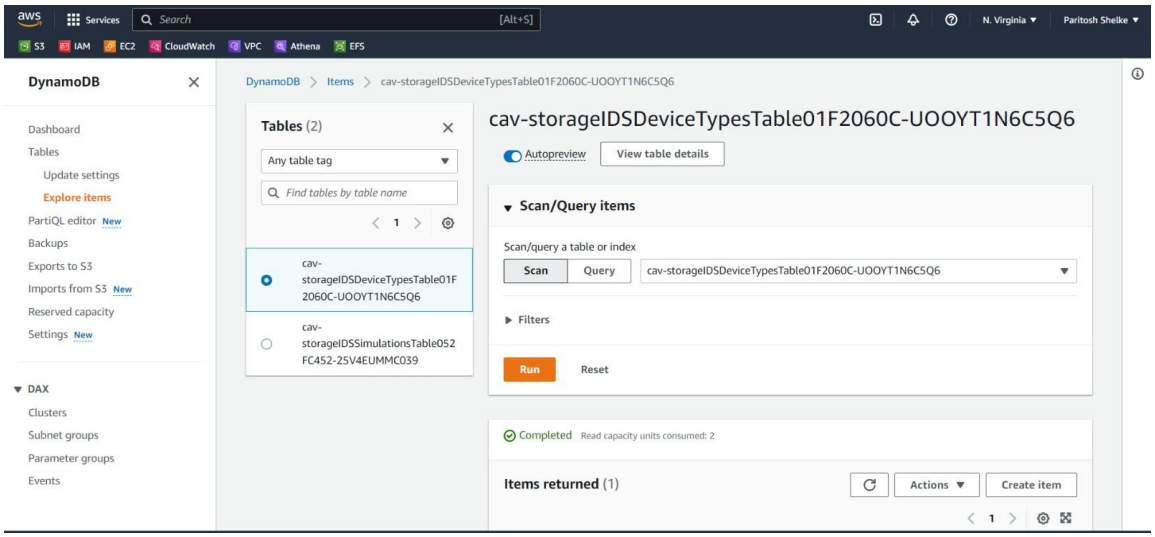

Figure 7: Table Results 3

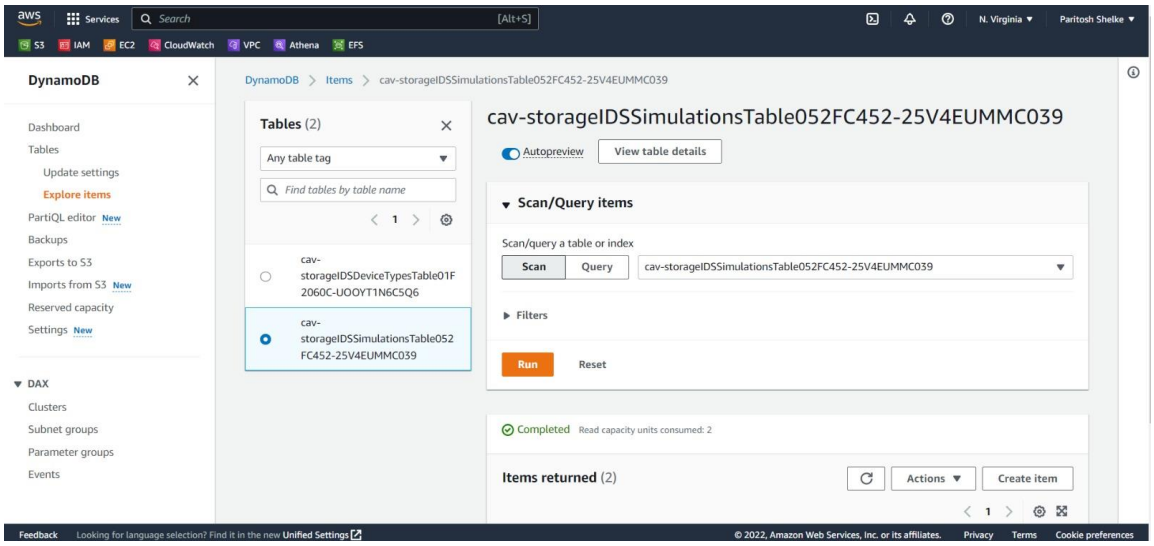

Figure 8: Table Results 4

IV. FUTURE SCOPE

The designed framework can be extended to various applications. The data being stored can be fed to AI/ML algorithms which can assist the vehicle in better perception of the environment surrounding it. On-board sensors like LiDAR, radar, and camera generate data that can be useful in making deep learning models which can facilitate the classification of objects on the path, enhancing the efficiency of the autonomous system. Furthermore, this data can also be utilized for remote diagnostics. As the data itself is hosted on AWS, it can be accessed by anyone in any part of the world. Thus, making it possible to diagnose any faults in the vehicle without needing to take the vehicle to the service station. Moreover, all this data can also be made available to the user with the aid of a mobile application. The user can track the performance and other vehicle-related data on the phone. As stated earlier, a data lake can also be created and hosted on the cloud [2]. As discussed, this data can be accessed by any software developer in any geographical location, contributing towards the development of the autonomous driving system.

V. REFERENCES

[1] C. Rec, “Building Connected Vehicle Solutions on the AWS Cloud,” Amazon, 08-Jan-2018. [Online]. Available: https://aws.amazon.com/blogs/iot/building-connected-vehicle-solutions-on-the-aws-cloud/. [Accessed: 08-Dec-2022].

[2] “Autonomous Vehicle Data Lake on AWS,” *Amazon AWS*. [Online]. Available: https://aws.amazon.com/automotive/solutions/autonomous-vehicle-data-lake/. [Accessed: 07-Dec-2022].