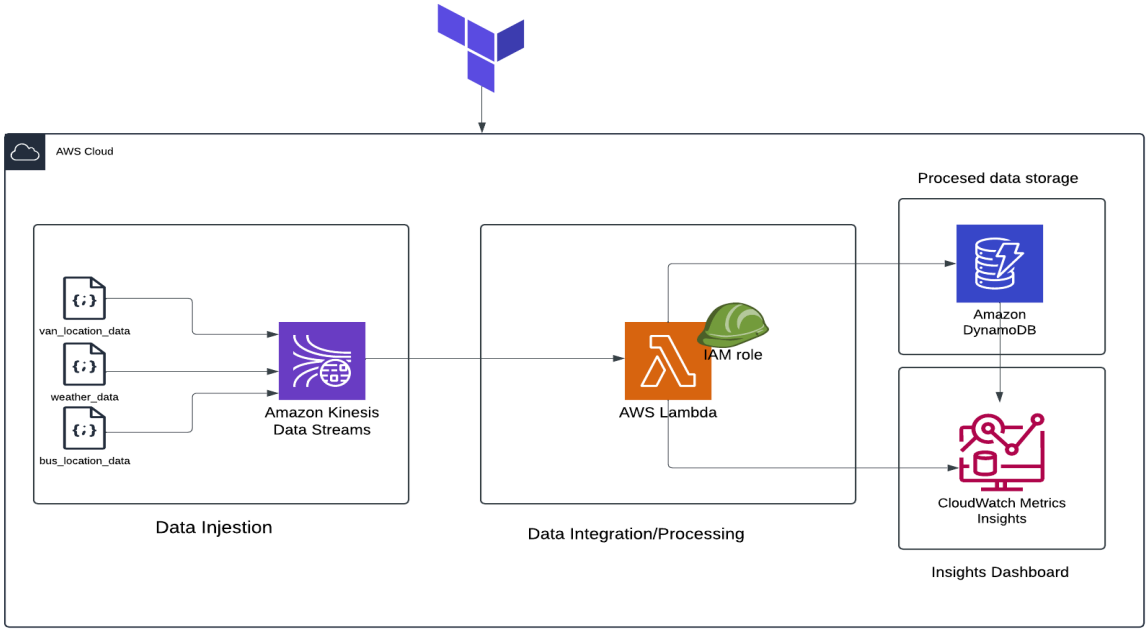
Project Report: Mobility Data Processing Pipeline (MDPP)

Overview

The Mobile Data Processing Pipeline (MDPP) employs a cloud native serverless approach to develop a real-time system designed to analyze bus locations and correlate them with weather data to detect transportation delays caused by adverse weather. The pipeline utilizes AWS services like Kinesis, Lambda, and DynamoDB to ensure scalability, real time processing and efficiency. The architecture is deployed on AWS using Infrastructure as code - Terraform.



AWS Kinesis Streams

Amazon Kinesis Data Streams is a fully managed, serverless data streaming service that stores and ingests various streaming data in real time at any scale, allowing you to easily write applications that process information in real-time. The collected data is available in milliseconds to allow real-time analytics use cases, such as real-time dashboards, real-time anomaly detection and analytics. By default, the data within the Kinesis Data Stream is stored for 24 hours with an option to increase the data retention to 365 days. In our project, three Kinesis streams ingest our data sources; *bus_location_data, weather_data* and *van_location_data* and stream a base64 encoded json object to a lambda function for processing.

AWS Lambda

In this project Lambda is used for processing data from Kinesis streams. A Lambda function *data_processing.py* reads the kinesis stream base64 encoded data, decodes it, parses JSON, and processes the data to extract insights and identify delays. Lambda is a serverless Compute managed Function as a Service that integrates the data and processes it to get insights. Serverless execution with Lambda simplifies infrastructure management and scales automatically with data

volume. Python was chosen for implementing the processing logic. The data_*processing.py* script handles data validation, data correlation, and delay identification.

AWS DynamoDB

Amazon DynamoDB is a serverless, NoSQL, fully managed database with single-digit millisecond performance at any scale horizontally. DynamoDB scales to workload changes as they ramp up or down, enabling data streaming use cases that can support any levels of demand. It provides fast and flexible NoSQL storage, ideal for querying processed data. Our processed data is stored in the *insights_table* DynamoDB table using the timestamps as identifiers.

Challenges and Solutions

1. Large Lambda Deployment Packages: The lambda function Deployment package exceeded the size limit for direct upload. To combat this, I deployed an S3 bucket, linking the Lambda function to the package in S3 with bucket versioning capability turned on.
2. Data Validation and Quality: Ensuring incoming data was complete and correctly formatted. Added validation logic in Lambda to filter incomplete or malformed records.
3. Base64 Decoding and JSON Parsing: Decoding and parsing base64-encoded data from Kinesis. Lambda decodes the base64 input and parses JSON using *json.loads*.

Future Work:

In a future work, Integrating advanced analytics and machine learning models to predict bus delays and optimize routes using historical and real-time data can enable proactive decision-making and improve transportation system efficiency. Developing user-friendly dashboards and web interfaces with AWS Amplify or AWS QuickSight for visualizing data insights in a clear and intuitive interface will improve stakeholder accessibility and enhance data-driven decision-making. Including security best practices in using .tfstate files.

Conclusion

The MDPP project uses AWS services to build a reliable, scalable, and efficient real-time data processing pipeline. Design choices ensure flexibility, ease of management, and scalability. Challenges such as large Lambda packages and data validation were resolved effectively. The system is designed to handle increased data loads, making it a robust solution for real-time analytics. By addressing these future work areas, the MDPP project can evolve into a more robust, scalable, and feature-rich platform.

References

AWS documentation
Terraform Documentation