Real-Time Cargo Tracking

&

Analytics System Proposal

Written by: Dahlton Seyler

06/29/24

Executive Summary

Based on my estimations and research, allocating more funds to our database service rather than the application infrastructure will be beneficial. Using Lambda/Cloud Run for hosting containers allows us to optimize cost and usage by managing each container individually.

The application should be divided into two main concerns: the API subscription endpoint and the real-time data display. This separation allows for independent scaling and cost optimization for each concern.

Using an image container service instead of a virtual machine vastly reduces the need for load balancing and full server maintenance. It also offers cost savings during periods of low requests, unlike virtual machines with fixed costs.

By opting for two purpose-built services instead of extensive cloud provider tools, we can avoid vendor lock-in and still benefit from their offerings. This approach ensures minimal downtime and changes if we switch providers or move to self-hosting.

This setup also supports organizational scalability. A single engineering team can manage multiple repositories, and as we grow, different teams can then own their repositories. Additionally, with appropriate languages and version control techniques, we can maintain a distributed monolith to leverage the benefits of both a monorepo and services.

Budget Estimate

Estimate Spreadsheet

	Estimated monthly budget
Lambda 1 - Subscriber Application	$6
Lambda 2 - Real-time dashboard Application	$6
Amplify - Static Site hosting	$0
API Gateway (1 million req/mo)	$4
Database - RDS	$100
Team - Size A	$27,500
Customer Subscription - First 3 months	$1,000
Total software cost	$1,116
Total per month	$28,616

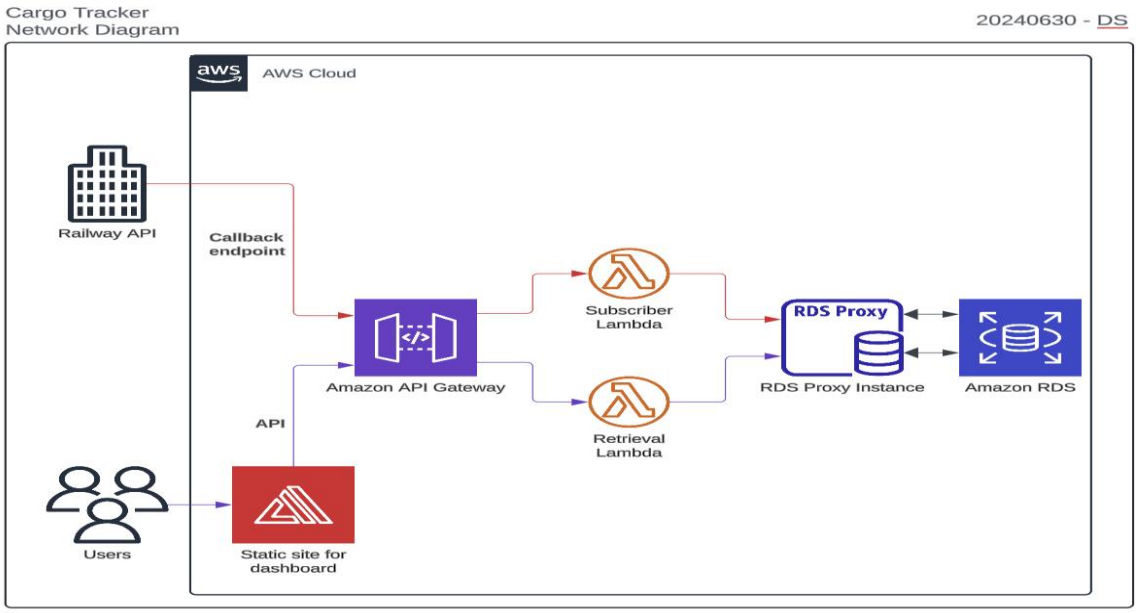
For the timeline, I estimate it will take several weeks to a few months to reach a usable product with a team of three. The team would consist of:

- **DevOps Engineer**: Responsible for ensuring that AWS and RDS are configured optimally, managing infrastructure as code, and maintaining efficient and reliable cloud services.
- **Backend Engineer**: Focused on developing the API subscription endpoint, handling data processing, integration with the database, and ensuring robust backend functionality.
- **Frontend Engineer**: Tasked with creating the real-time data display, implementing user interfaces, and ensuring a seamless user experience.

This team structure ensures that each critical area is covered, promoting efficient development and deployment processes.

Network Diagram

Lucidchart



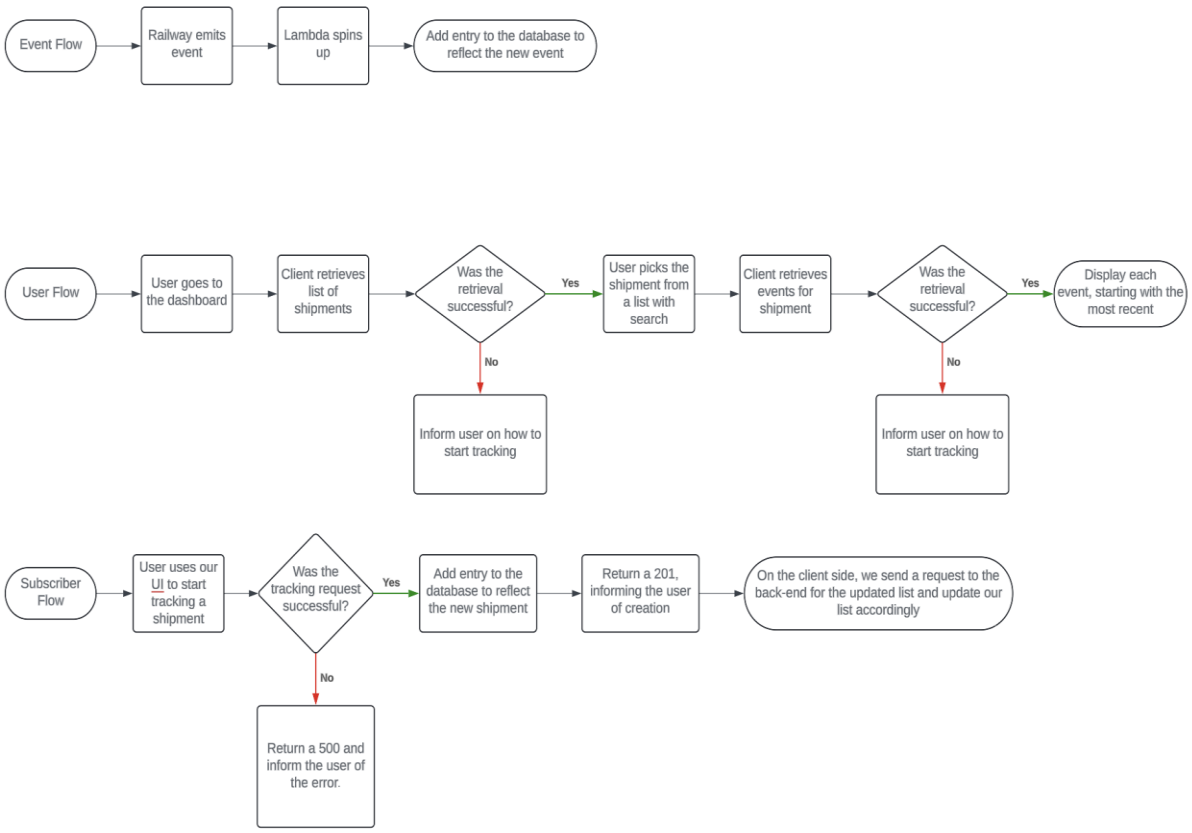
For my network diagram and subsequent application architecture, I have chosen to use AWS, leveraging their API Gateway, Lambda, RDS, and Amplify technologies. This architecture will be cost-effective for getting us up and running efficiently.

- **API Gateway and Lambda**: This combination provides high availability, resilience, and quick recovery. It allows for automatic horizontal scaling within predefined budgets.
- **RDS**: The database will handle our needs for some time before requiring vertical or horizontal scaling.
- **Amplify**: This service will streamline deployment and ensure that the latest changes are automatically deployed.

This setup ensures a smooth deployment process and robust scalability, aligning with our cost and performance goals.

Linear Business Process

Lucidchart



For my business processes, I chose to keep it simple. Events will be fired by the railroad API, and we will read and store them. When a user wants to see the information, we will retrieve it from the database and display it.

When a user wishes to subscribe to a new shipment, we will add a new page displaying all relevant events with options to cancel or delete the shipment.

Risks

DDOS attacks are always possible, mitigation strategies may include:

- Queueing, we can use a service such as SQS to ensure that API calls are handled at a steady rate.
- Budget limits, we can set our own limits for our app to have a hard stop when certain criteria are met.

Sensitive data:

- The data we are dealing with here is very akin to tracking numbers, which are for the most part not sensitive data as there are no real addresses shown.