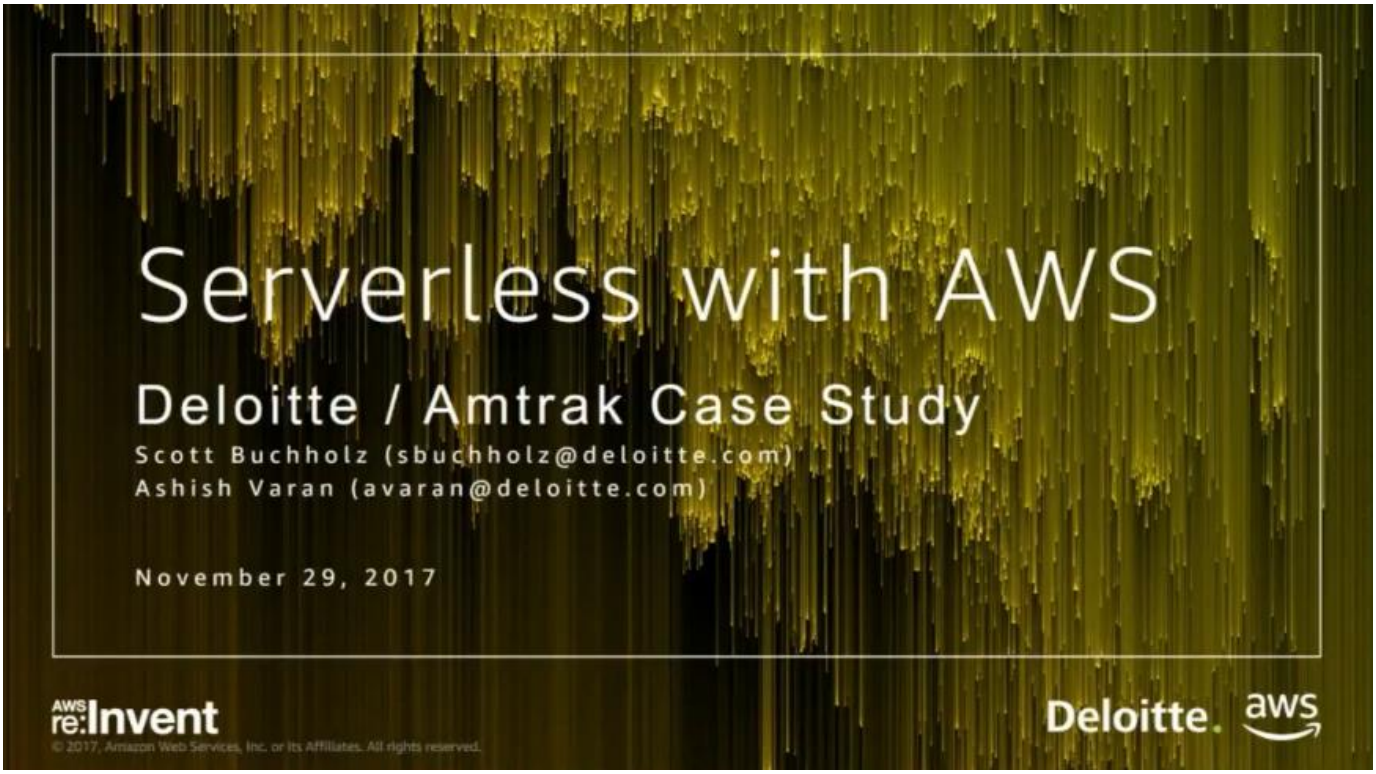



As the nation's only high-speed intercity passenger rail provider, Amtrak needs to know critical information to run their business such as: Who’s onboard any train at any time? How are booking and revenue trending? Amtrak was faced with unpredictable and often slow response times from existing databases, ranging from seconds to hours; existing booking and revenue dashboards were spreadsheet-based and manual; multiple copies of data were stored in different repositories and Mainframes, lacking integration and consistency; and operations and maintenance (O&M) costs were relatively high. Join us as we demonstrate how Deloitte and Amtrak successfully went live with a cloud-native operational database and analytical datamart for near-real-time reporting in under six months. We highlight the specific challenges and the modernization of architecture on an AWS native Platform as a Service (PaaS) solution. The solution includes cloud-native components such as AWS Lambda for microservices, Amazon Kinesis and AWS Data Pipeline for moving data, Amazon S3 for storage, Amazon DynamoDB for a managed NoSQL database service, and Amazon Redshift for near-real time reports and dashboards. Deloitte’s solution enabled “at scale” processing of 1 million transactions/day and up to 2K transactions/minute. It provided flexibility and scalability, largely eliminate the need for system management, and dramatically reduce operating costs. Moreover, it laid the groundwork for decommissioning legacy and Mainframe systems, anticipated to save at least $1M over 3 years.


The system we are talking about today is a mission critical component of running Amtrak, we need to know who we have on the trains, even when people get on and off at different points in time. This is more logistically difficult than airplane passenger tracking. We have monitoring using AppDynamics. It also helps the sales and marketing team know who and how many people are getting on the trains, there is a booking and ticket modules in this system.

# INITIAL SOLUTION ASSESSMENT

We analyzed the current system, generated several options and provided a recommended path forward

| Option | Analysis Dimensions: Implementation Cost | Implementation Complexity | Time to Market | Operational Simplification | Enabling Real-Time Reporting | Single system? |
|---|---|---|---|---|---|---|
| Hybris | ◑ | ◕ | ◔ | ◑ | ◕ | No |
| SAP R/3 (Current) | ◔ | ◔ | ◔ | ◑ | ◔ | No |
| SAP S/4 (HANA + potential 3rd party bolt on) | ○ | ○ | ○ | ● | ● | Yes |
| Other COTS Products (e.g., Revenue Accounting System) | ◔ | ◔ | ○ | ◔ | ◕ | TBD |
| ARROW Workaround | ● | ● | ● | ○ | ○ | No |
| Sales ODS (minimal change) | ◕ | ◕ | ◕ | ◔ | ○ | No |
| Sales ODS (refactor) | ◑ | ◑ | ◕ | ◕ | ◕ | Yes |

**Guiding Principles:**

- Enable real-time dashboards/reporting (i.e., sales and marketing, revenue analytics and fraud)
- Simplify and consolidate data storage
- Maximize agility and performance
- Simplify operational environment and minimize operational management
- Isolate downstream systems from direct access
- Minimize cost

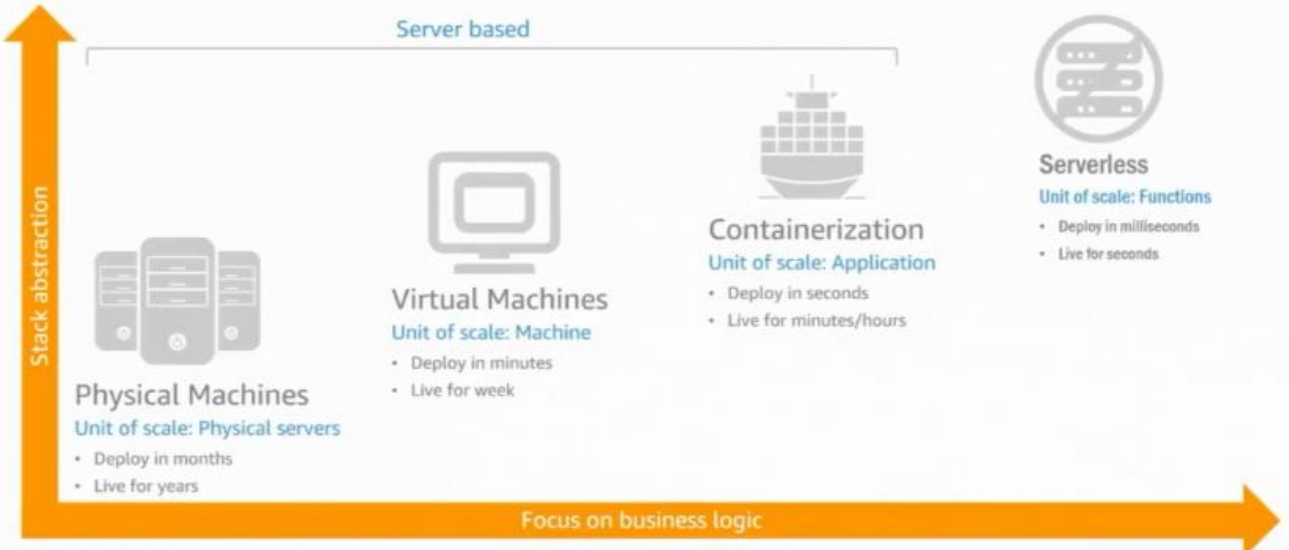
**Recommendation:**

- Explore AWS Serverless services in a Proof of Concept





# ROAD TO SERVERLESS SOLUTIONS



# SERVERLESS: SUITABLE USE CASES

Serverless is not a one size fits all solution and cannot replace all on premise applications

**USE CASES MOST SUITED FOR SERVERLESS**

- ✓ **Short running computational tasks**
- ✓ **Web applications**
- ✓ **Real-time analytics and data processing**
- Mobile backend
- IoT backend

**USE CASES LEAST SUITED FOR SERVERLESS**

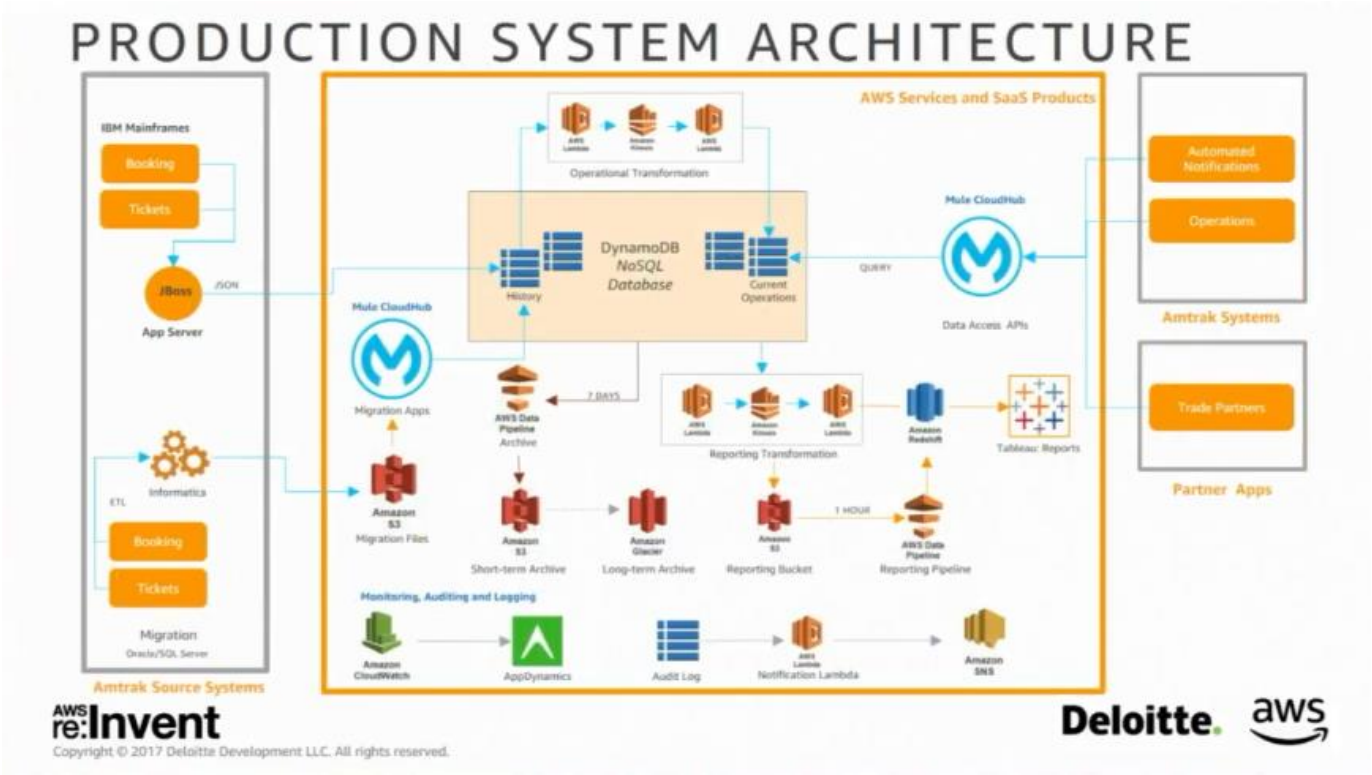
- Legacy applications
- Long-running computational tasks
- Applications requiring command line server access
- Applications requiring significant storage or memory



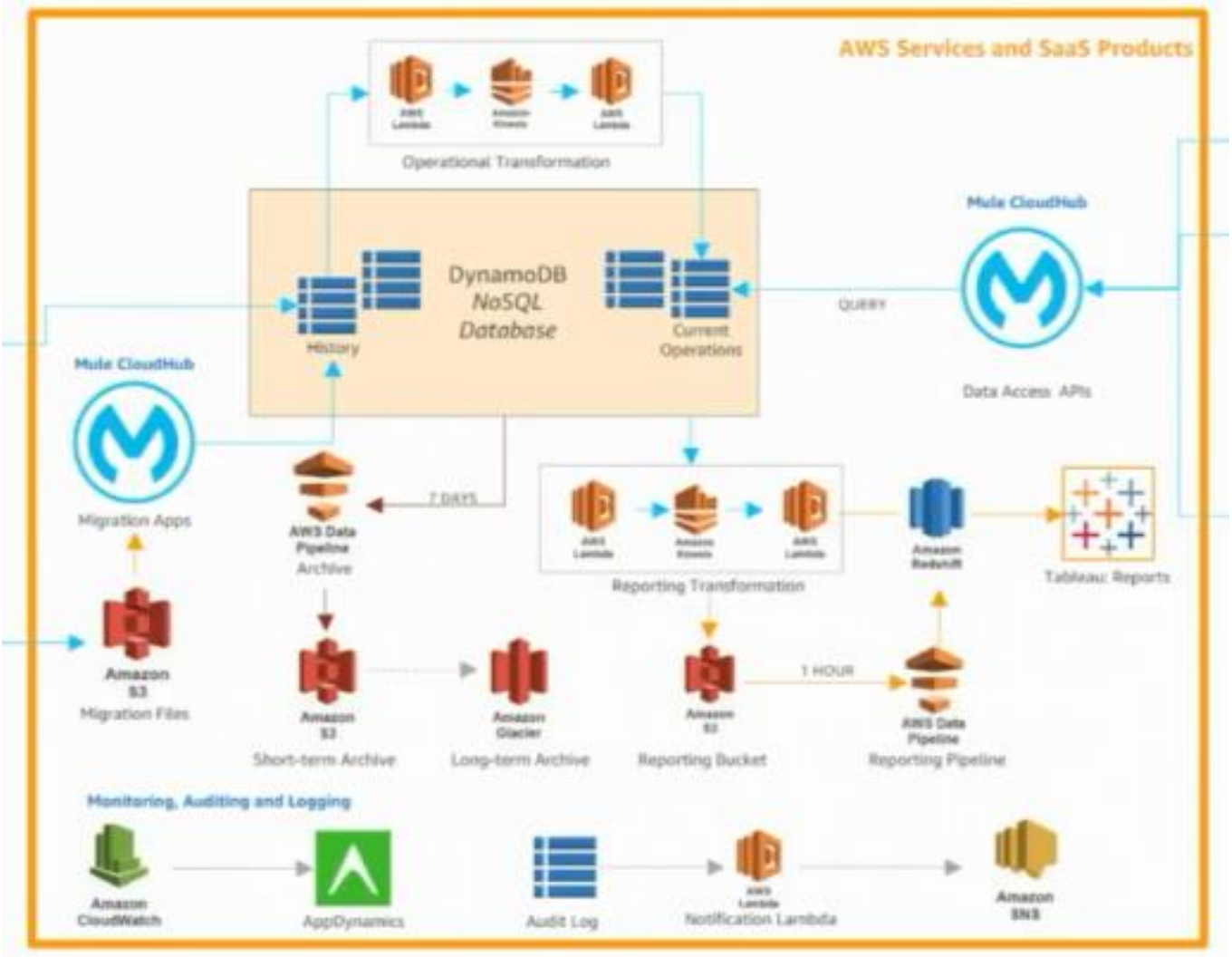


# SOLUTION SERVICES OVERVIEW

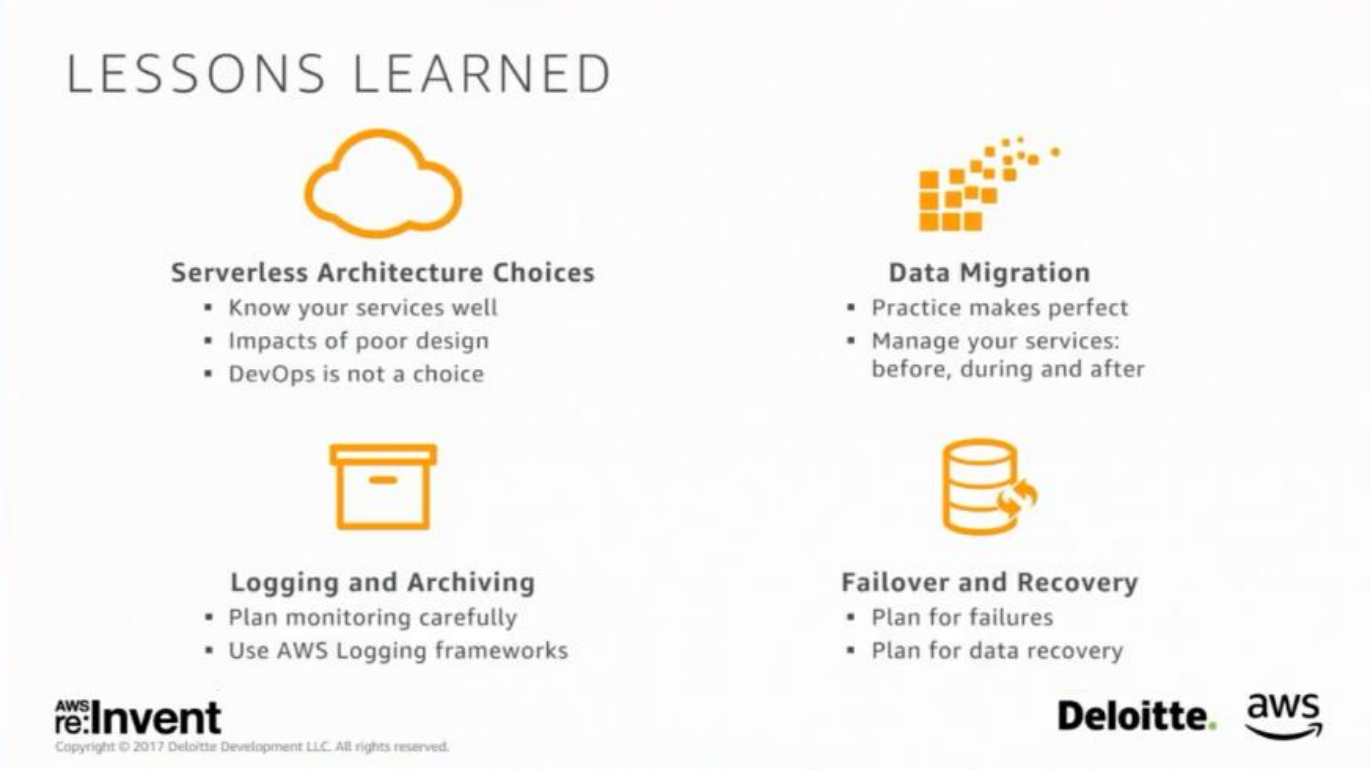
The solution was built on AWS's serverless cloud-computing services



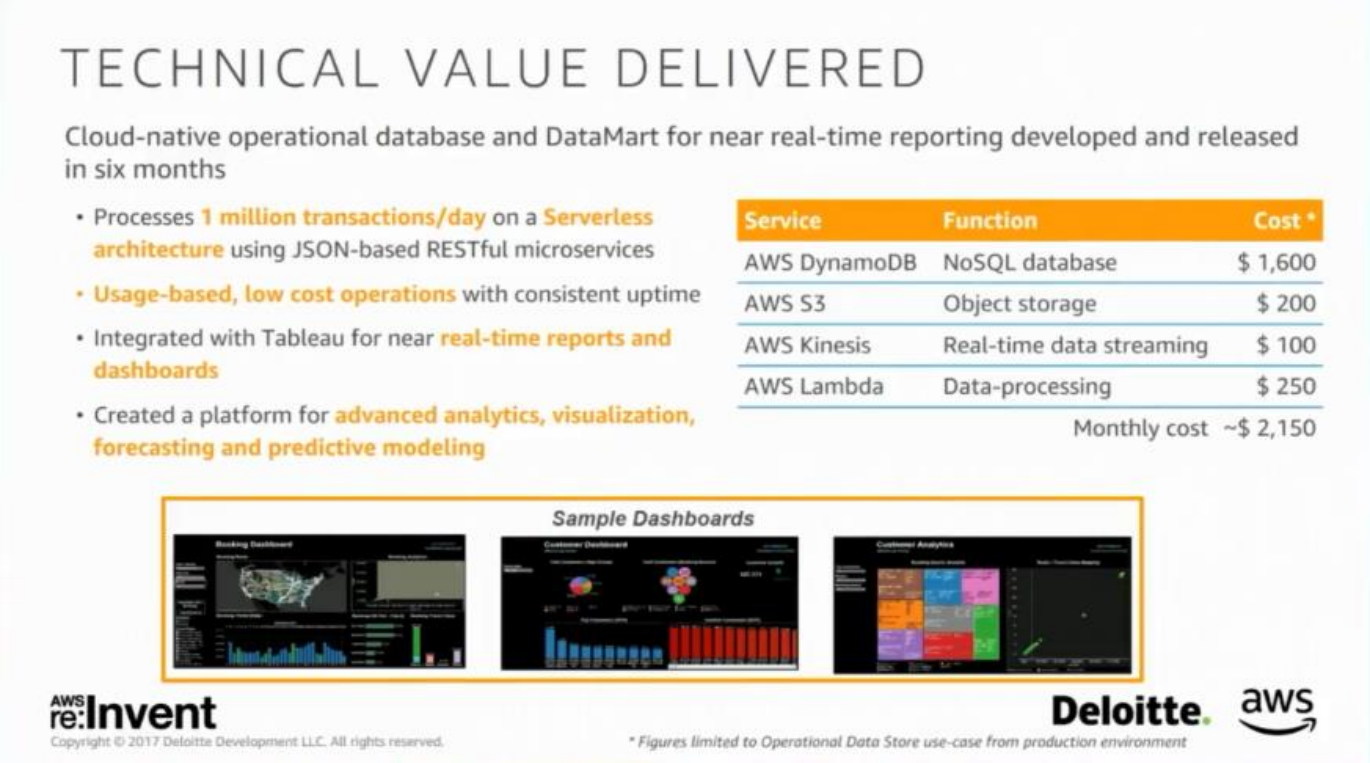

On the ***left*** you have legacy transactional systems for Booking and Tickets systems that are collecting data from the mainframes, on the right you have all the downstream systems for Operations and notifications and travel alerting systems that integrate with all the trade partners who can also access it. The middle portion is what was built in this project.



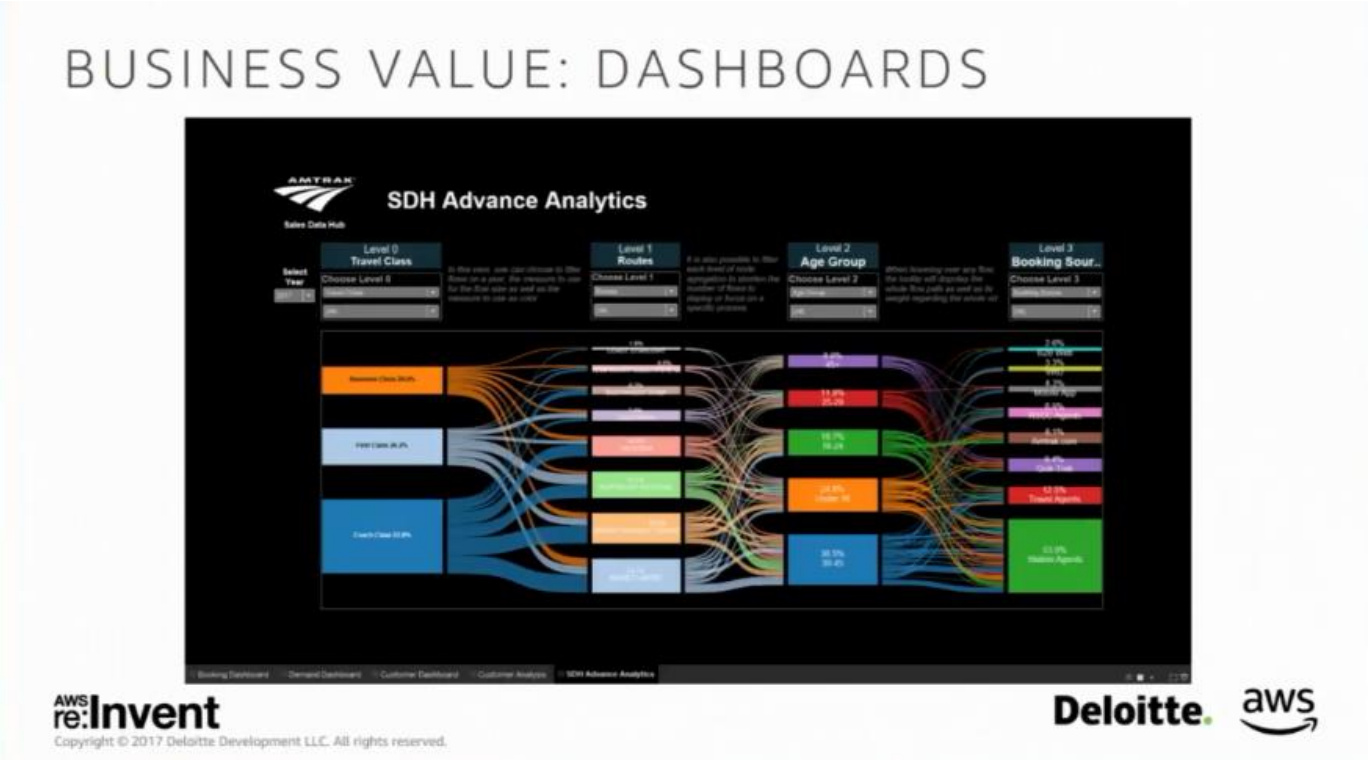

There are DynamoDB tables that collect all the tickets data, we use Kinesis to move data around and cleanse the data using Lambda, S3 and Glacier are used for archiving. We use a set of lambdas that move the data to a data pipeline and from there on to Redshift. We also had to build custom logging functions to capture several business events or errors within the system, and also use lambdas for sending emails and notifications to different tiers in the system



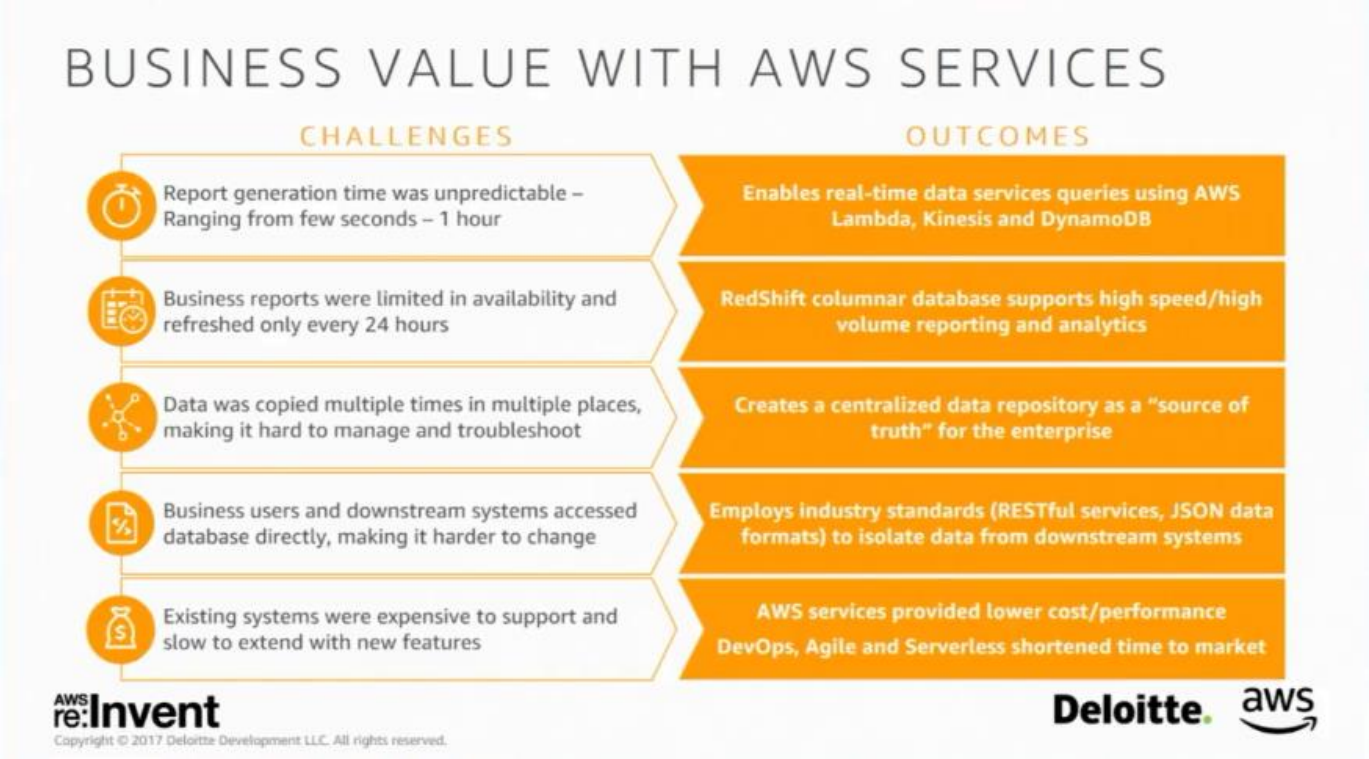

DynamoDB allows for up to 5 keys, so you have to choose your indexes carefully particularly when choosing your sort keys. There is a limit of 400KB for your records in DynamoDB, you need a strategy for handling payloads over 400KB. For Kinesis you have to optimize the number of shards you have in Kinesis so that they don't take longer time before being picked up before deletion in Kinesis. You have to build efficient JSON transformations when needed. Store your data in S3 and use custom jobs to load data back into DynamoDB when needed, you also have to change your read and write capacities when loading data and remember to change it back when done.




This system was built in about 6 months from concept to production, we also have a couple of Tableau dashboards for business consumers.




Above is one of the available dashboards from Tableau. It is a flow diagram that allows you to see the relationships between different elements by being able to move the columns around if you are in the Amtrak sales and marketing team.




AWS re:Invent
THANK YOU!
AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Deloitte.
aws