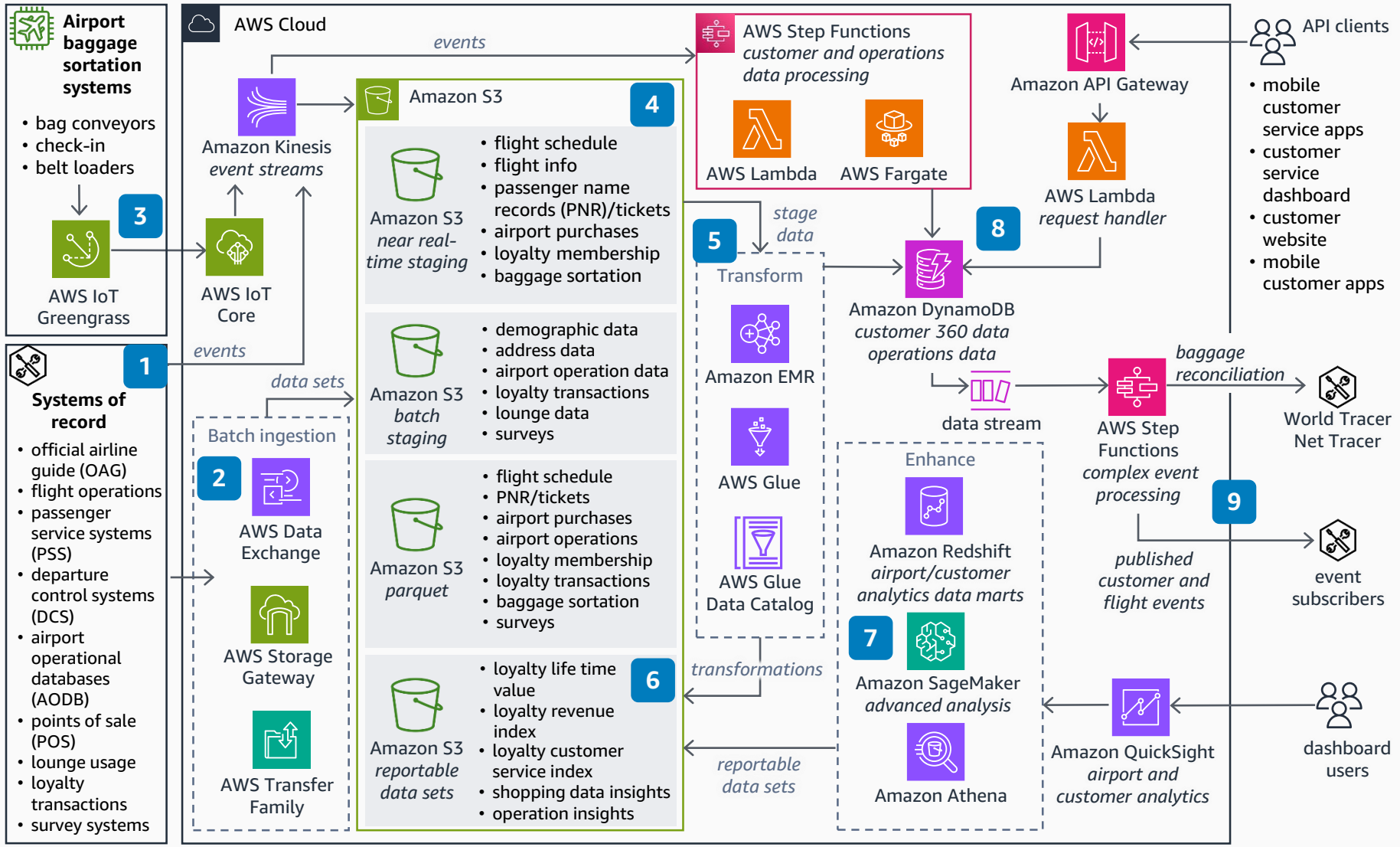
Guidance for Airport Data Management on AWS

Enhanced Traveler Experience

This diagram shows how to build a data management system that will generate meaningful insights with information from airlines and vendors to enhance the traveler experience.



1. Build an operational data management system with systems of record such as schedules, near real-time flight information, traveler check-ins, loyalty transactions, lounge usage, and transactions in the airport.
2. Collect addresses, demographics, and other public data sets with **AWS Data Exchange**. Ingest private data sets with **AWS Storage Gateway** and **AWS Transfer Family**.
3. Optionally, enhance the baggage system with a radio-frequency identification (RFID) type of sortation and tracking system. Or, leverage the bar code scanners and baggage sortation events. Combine **AWS IoT Greengrass**, **AWS IoT Core**, and **Amazon Kinesis** to ingest sortation events.
4. Provide staging for ingesting all batch and near real-time data using cost-effective storage classes in **Amazon Simple Storage Service** (Amazon S3).
5. Use **Amazon EMR** and **AWS Glue** to transform your data. Use open standards to build the data lake using the same data as the operational data management system. Use a read pattern schema to make the raw data and curated data readily available for all user roles.
6. Build all reportable data sets in **Amazon S3**, and leverage **Amazon Redshift**, **Amazon Athena**, and **Amazon QuickSight** for analytics. Optionally, build data marts in **Amazon Redshift** for heavily used analytics. For miscellaneous requirements, publish the AWS Glue Data Catalog, and use **Athena** for analysis using the data lake built in Step 5.
7. Use **Amazon SageMaker** to provide standard artificial intelligence and machine learning (AI/ML) models for operational analytics. You can also use **SageMaker** to build your own models on top of the data.
8. Use purpose-built databases like **Amazon DynamoDB**, and serverless services like **AWS Lambda** and **Amazon API Gateway**, to deliver microservices and events for operational data stores. Build near real-time operational dashboards and customer applications using these microservices.
9. Leverage **Amazon DynamoDB Streams** and **AWS Step Functions** to publish flight and customer events to downstream systems, like baggage reconciliation and ground transportation.

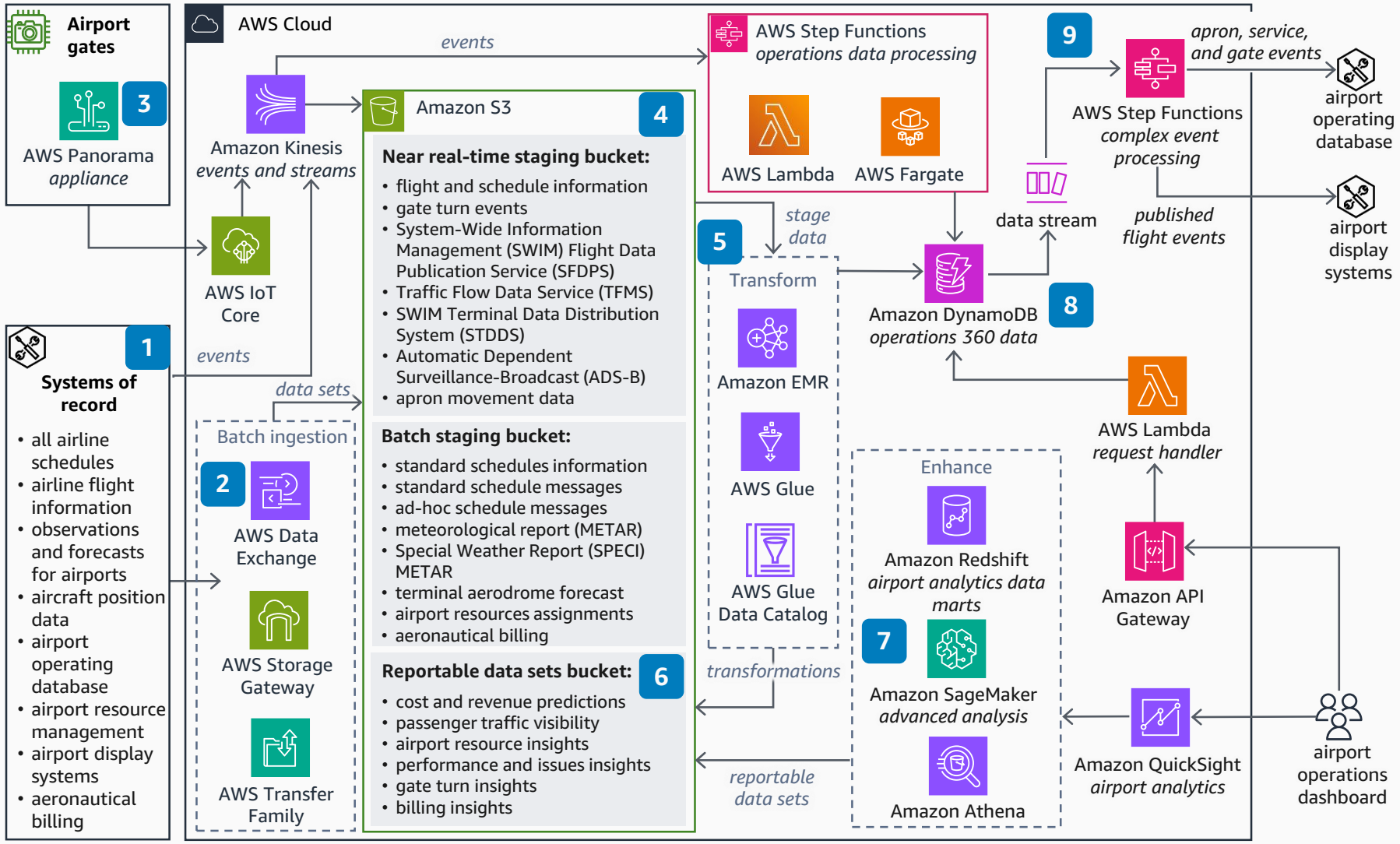
Reviewed for technical accuracy June 16, 2023

AWS Reference Architecture

Guidance for Airport Data Management on AWS

Optimizing Operations

This diagram shows how to build a data management system to monitor airport operations in near real-time. It can be used to predict costs, revenue, turnaround times, and potential delays using open data standards, purpose-built databases, and an extensive serverless architecture.



1. Build the data management system with systems of record such as airline schedules, near real-time flight information, weather forecasts, aircraft position data, airport resources, and billing.
2. Leverage **AWS Data Exchange** to create the data collection from public sources such as the weather and aircraft position. Ingest private data sets with **Storage Gateway** and **Transfer Family**.
3. Use **AWS Panorama** or follow Aircraft Turn Tracking to passively collect and leverage aircraft gate turn events.
4. Provide staging for ingesting all batch and near real-time data using cost-effective storage classes in **Amazon S3**.
5. Transform your data with **Amazon EMR** and **AWS Glue**. Use open standards to build a data lake using the same data as the operational data management system. Use a read pattern schema to make the raw data and curated data readily available for all roles.
6. Build all reportable data sets in **Amazon S3**. Build data marts in **Amazon Redshift** for heavy analytics. For miscellaneous requirements, publish the Data Catalog and use **Athena** and **QuickSight** for analysis using the data lake.
7. Use **SageMaker** to provide standard AI/ML models for operational analytics, or use **SageMaker** to build your own models on top of the data.
8. Use purpose-built databases like **DynamoDB** and serverless services like **Lambda** and **API Gateway** to deliver microservices and events for operational data stores. Build a near real-time operational dashboard and customer applications leveraging these microservices.
9. Leverage **DynamoDB Streams** and **Step Functions** to publish flight and aircraft movement events to keep the Airport Operating Database and Airport Display systems current.

Reviewed for technical accuracy June 16, 2023

AWS Reference Architecture