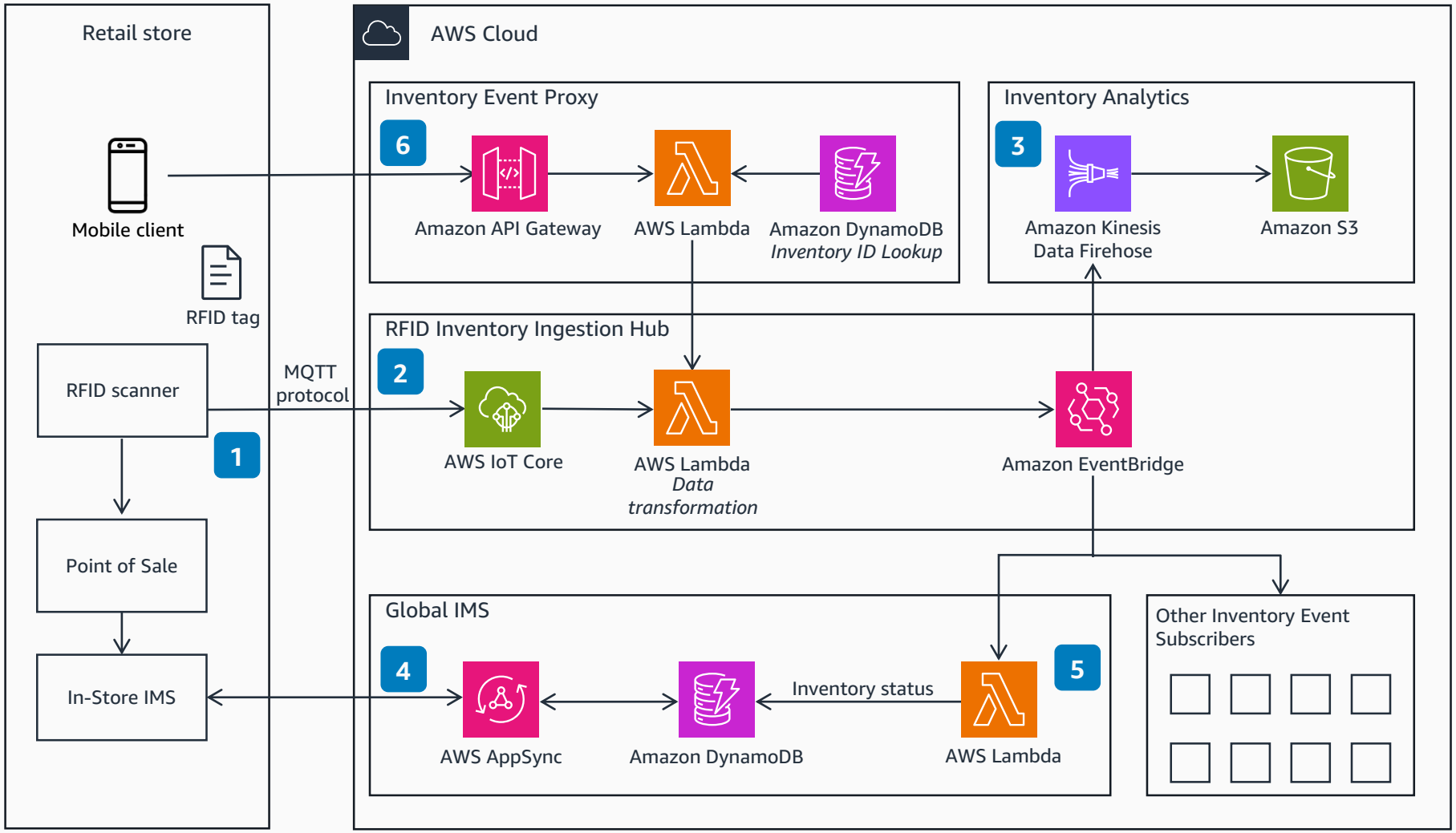
Guidance for RFID Store Inventory on AWS

Managing Store Inventory using RFID

This is a logical architecture showing RFID integration of retail store inventory systems using AWS services. It shows the foundational components and flows needed to monitor, maintain, and act on product inventory data.



1. Inventory items scanned in the store are sent to **AWS IoT Core** in the Inventory Ingestion Hub using the MQTT protocol. In-store RFID scanners can also link to in-store Point of Sale and inventory management systems (IMS).

2. The Inventory Ingestion Hub handles ingestion of inventory scan events. **AWS IoT Core** uses **AWS Lambda** for data transformation tasks before being published to **Amazon EventBridge**.

3. The Inventory Analytics layer reads all events from **EventBridge. Amazon Kinesis Data Firehose** loads data to **Amazon Simple Storage Service (Amazon S3)** for analytics and machine learning (ML) use cases, such as store replenishments.

4. The Global IMS subscribes to **EventBridge** to maintain near real-time inventory updates in **Amazon DynamoDB**. This occurs by triggering the **AWS Lambda** function to update the **DynamoDB** table and perform transformation, if applicable. As updates take place, **AWS AppSync** shares them back to the In-Store IMS for reconciliation.

5. **EventBridge** posts events to other enterprise systems registered as event targets based on defined rules.

6. The Inventory Event Proxy accepts mobile scans, such as a quick response (QR) code and near field communications (NFC) through an **Amazon API Gateway** with **Lambda** as the backend. The event is matched to an RFID tag stored in **DynamoDB** and passed to the Inventory Ingestion Hub for processing.

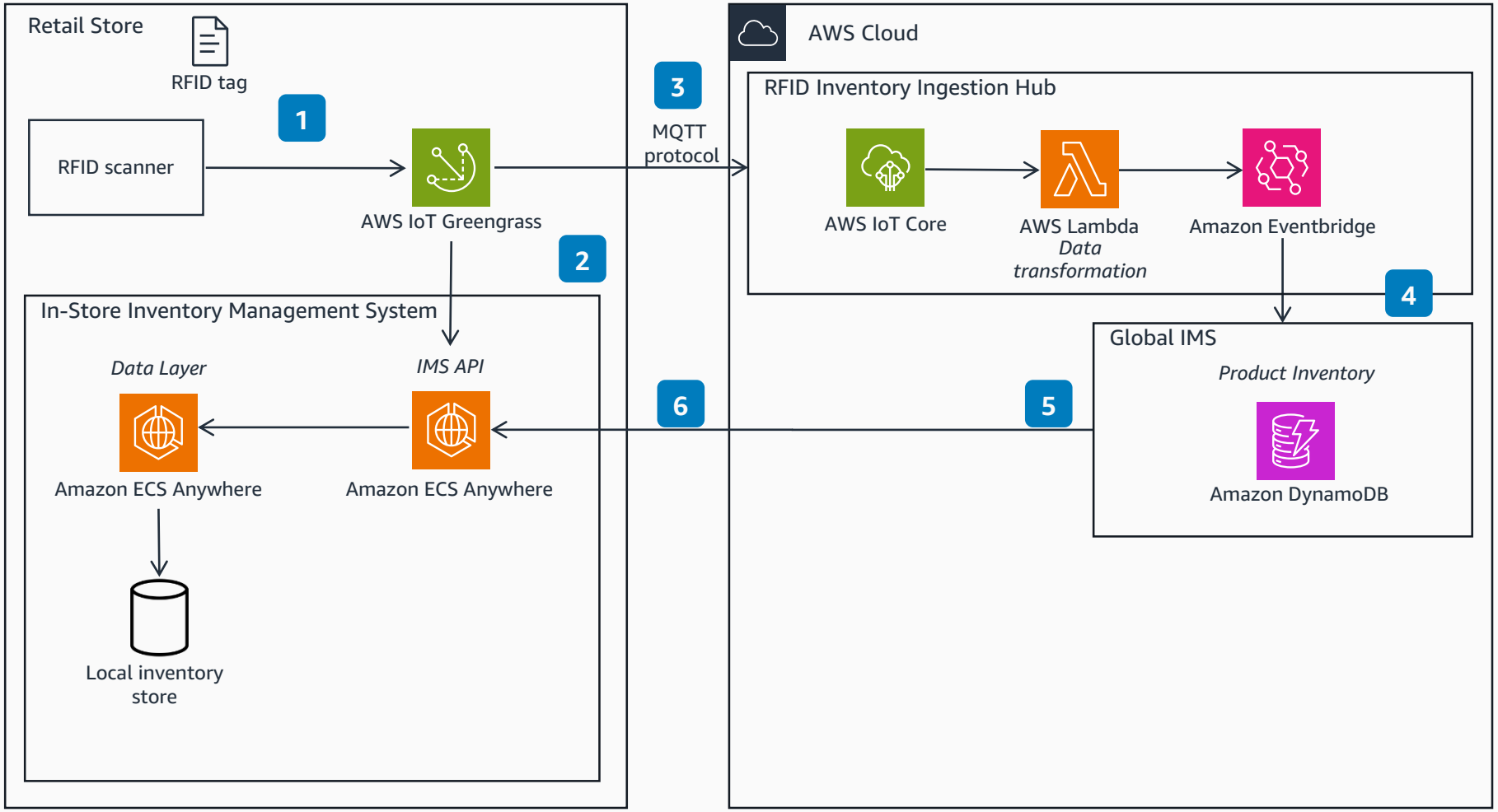
Reviewed for technical accuracy June 7, 2023

AWS Reference Architecture

Guidance for RFID Store Inventory on AWS

Counting Store Inventory using RFID

This is a logical end-to-end architecture for RFID integration in a retail store that uses AWS services to perform inventory reconciliation.



1. You can quickly audit inventory by scanning inventory both stocked on shelves and hung on racks. Scanned tags are sent to **AWS IoT Greengrass** for pre-processing and de-duplication through **Lambda**. The event is published to **EventBridge**.

2. **AWS IoT Greengrass** notifies the In-Store IMS of the inventory scans. The In-Store IMS runs on **Amazon Elastic Container Service (Amazon ECS) Anywhere** to allow for centralized management and deployment of updates. It comprises of a data layer and API layer. The In-Store IMS will also house a local inventory storage to have inventory data locally.

3. **AWS IoT Greengrass** sends the scanned inventory IDs and the store location to the Inventory Ingestion Hub. The Inventory Ingestion Hub uses **AWS IoT Core** to receive data from the edge, have the event-driven backend of **Lambda** perform data transformation if applicable, and publish to **EventBridge**. **EventBridge** then publishes this event to subscribers or downstream applications.

4. The Global IMS subscribes to the inventory scan events in the Inventory Ingestion Hub. This allows it to reconcile any inventory discrepancies into the product inventory database hosted on **DynamoDB**.

5. Once the inventory audit is complete, the Global IMS returns any updates or notifications back to the In-Store IMS, which completes the asynchronous update loop.

6. Inventory discrepancies are reconciled in the In-Store IMS, which then updates the Global IMS to help ensure that both systems are synced.

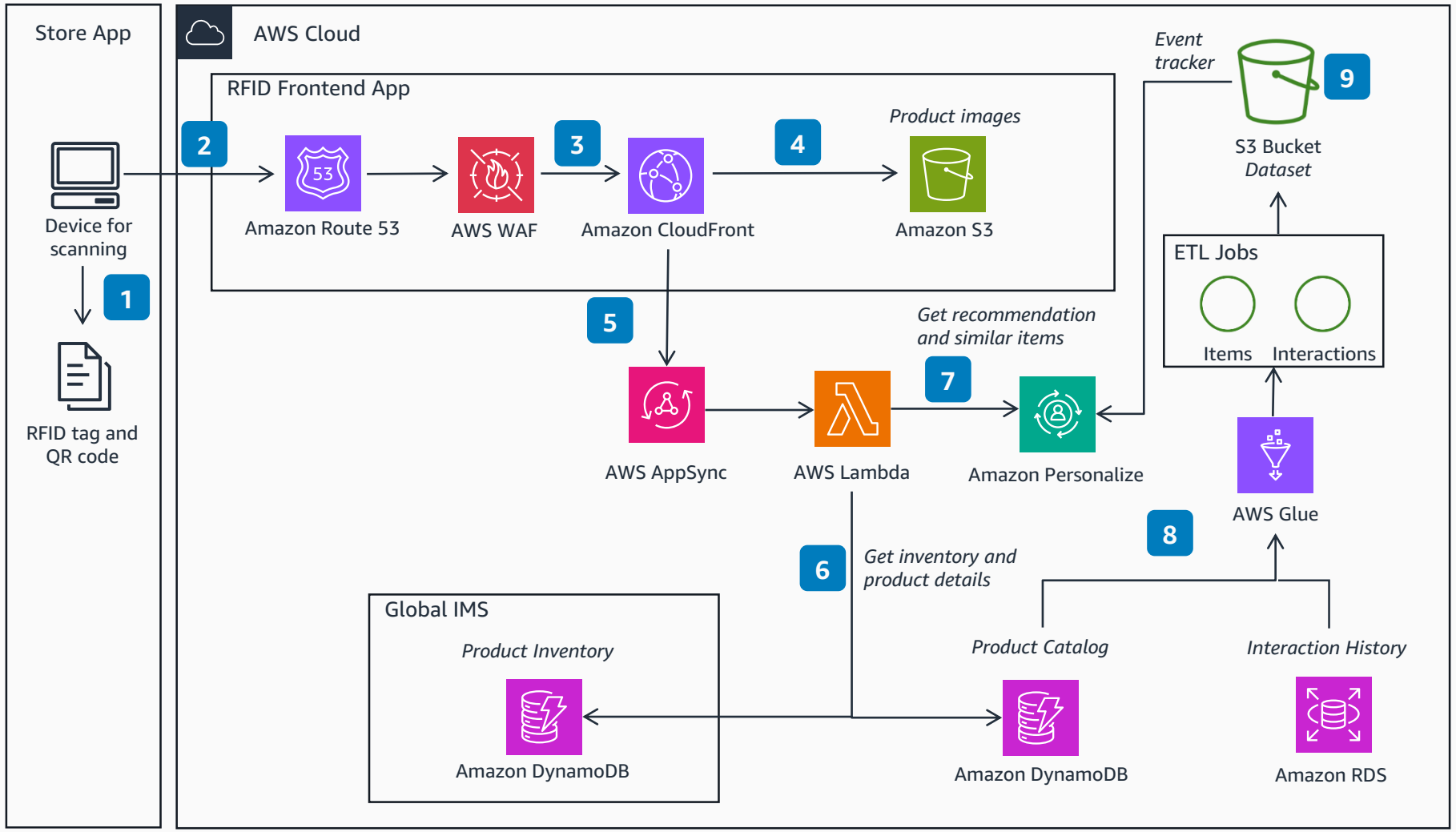
Reviewed for technical accuracy June 7, 2023

AWS Reference Architecture

Guidance for RFID Store Inventory on AWS

Identifying Inventory using RFID

This is a logical end-to-end architecture for RFID integration in a retail store that uses AWS services when product details and recommendations need to be returned to a customer.



1. A store user scans products of interest using the Store App on their device.
2. The Store App makes HTTPS requests to **Amazon Route 53** to provide domain name service (DNS) translation.
3. The request is routed to the nearest **Amazon CloudFront** edge location. **AWS WAF [Web Application Firewall]** rules are applied to traffic to protect against exploits.
4. The static website and assets (such as HTML, image, and video) stored in **Amazon S3** are returned.
5. **AWS AppSync** handles queries by routing to resolvers, such as **Lambda**.
6. **Lambda** uses ProductID to return detailed product information, such as size, color, options, characteristics, and current inventory stored in **DynamoDB.**
7. **Lambda** uses ProductID to return recommendations from **Amazon Personalize** for items frequently purchased together.
8. **AWS Glue** crawls the Product Catalog and Interaction History and creates a data catalog stored in **Amazon S3,** where ETL jobs can transform data to support **Amazon Personalize** training jobs.
9. Datasets are added to an **S3** bucket, and the training cycle is initiated in **Amazon Personalize**.

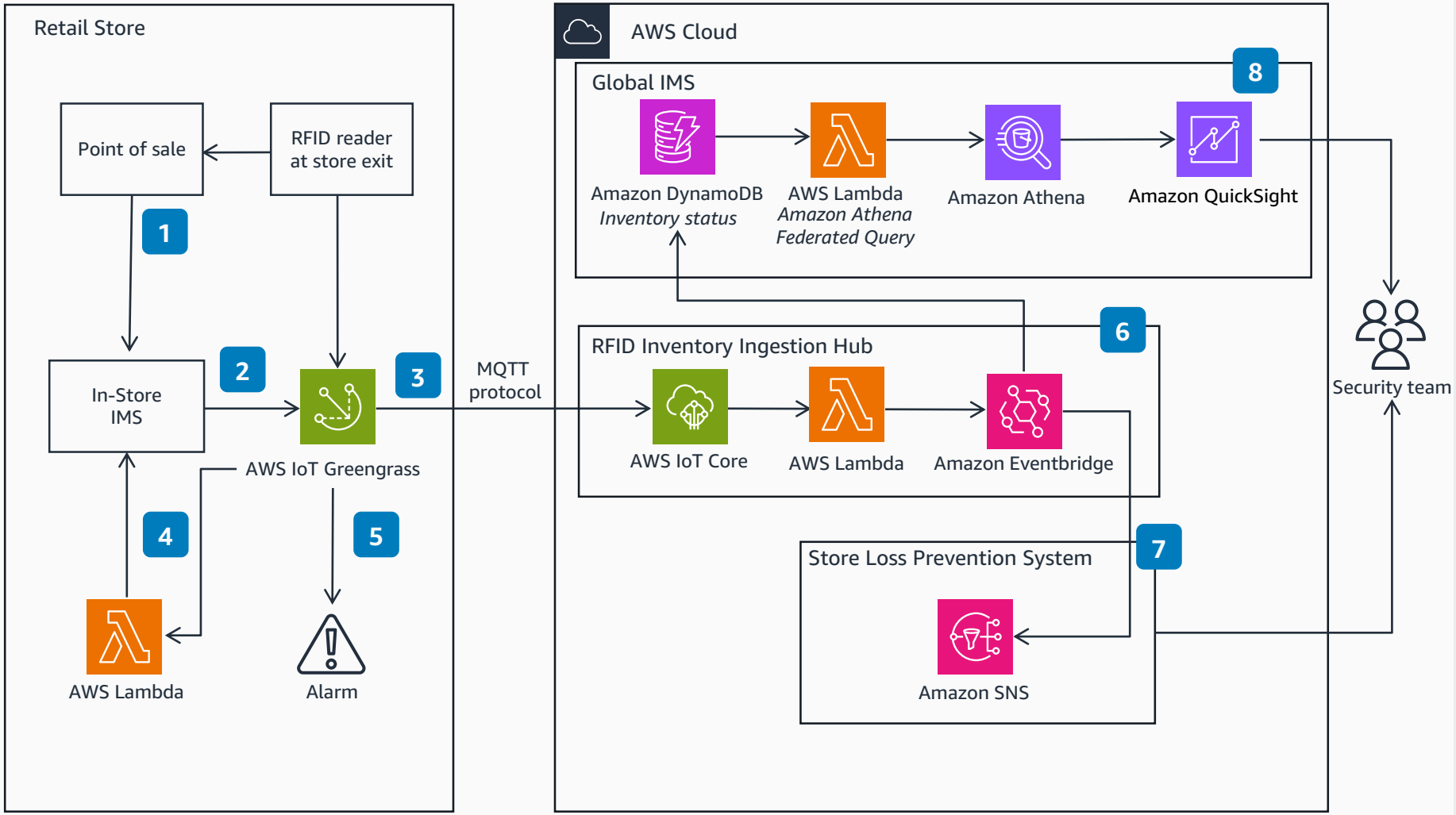
Reviewed for technical accuracy June 7, 2023

AWS Reference Architecture

Guidance for RFID Store Inventory on AWS

Detecting Shrink using RFID

This is a logical end-to-end architecture for RFID integration in a retail store that uses AWS services to prevent and detect asset loss.



1. Completed purchases at the Point of Sale sends an update to the In-Store IMS to update inventory status.
2. The In-Store IMS updates the Global IMS asynchronously to keep the global inventory status current. This uses the RFID tag to associate the purchase to a specific item.
3. As an RFID tag passes through an RFID reader at the store exit, event data is sent to **AWS IoT Core.**
4. A **Lambda** function packaged within **AWS IoT Greengrass** is initiated when an asset is passed by the RFID reader. The function validates if the tag is part of a purchase event. The logic will check the RFID tag and compare it to the In-Store IMS.
5. If the item has not yet been purchased, **AWS IoT Greengrass** triggers the device alarm in the physical store and then sends the event to the Inventory Ingestion Hub for further processing.
6. The Inventory Ingestion Hub handles ingestion of inventory scan events. **AWS IoT Core** and **Lambda** publish the events to **EventBridge**.
7. The Store Loss Prevention System picks up the event from the Inventory Ingestion Hub and sends SMS messages through **Amazon Simple Notification Service (Amazon SNS)** to notify proper personnel.
8. The Global IMS is updated with the events. Proper teams can review using **Amazon QuickSight**, which generates reports and visualizations from data that has been queried in **Amazon Athena**. **Athena** has a direct connection to the inventory in **DynamoDB** through a **Lambda** function that runs federated queries against the inventory.

Reviewed for technical accuracy June 7, 2023

AWS Reference Architecture