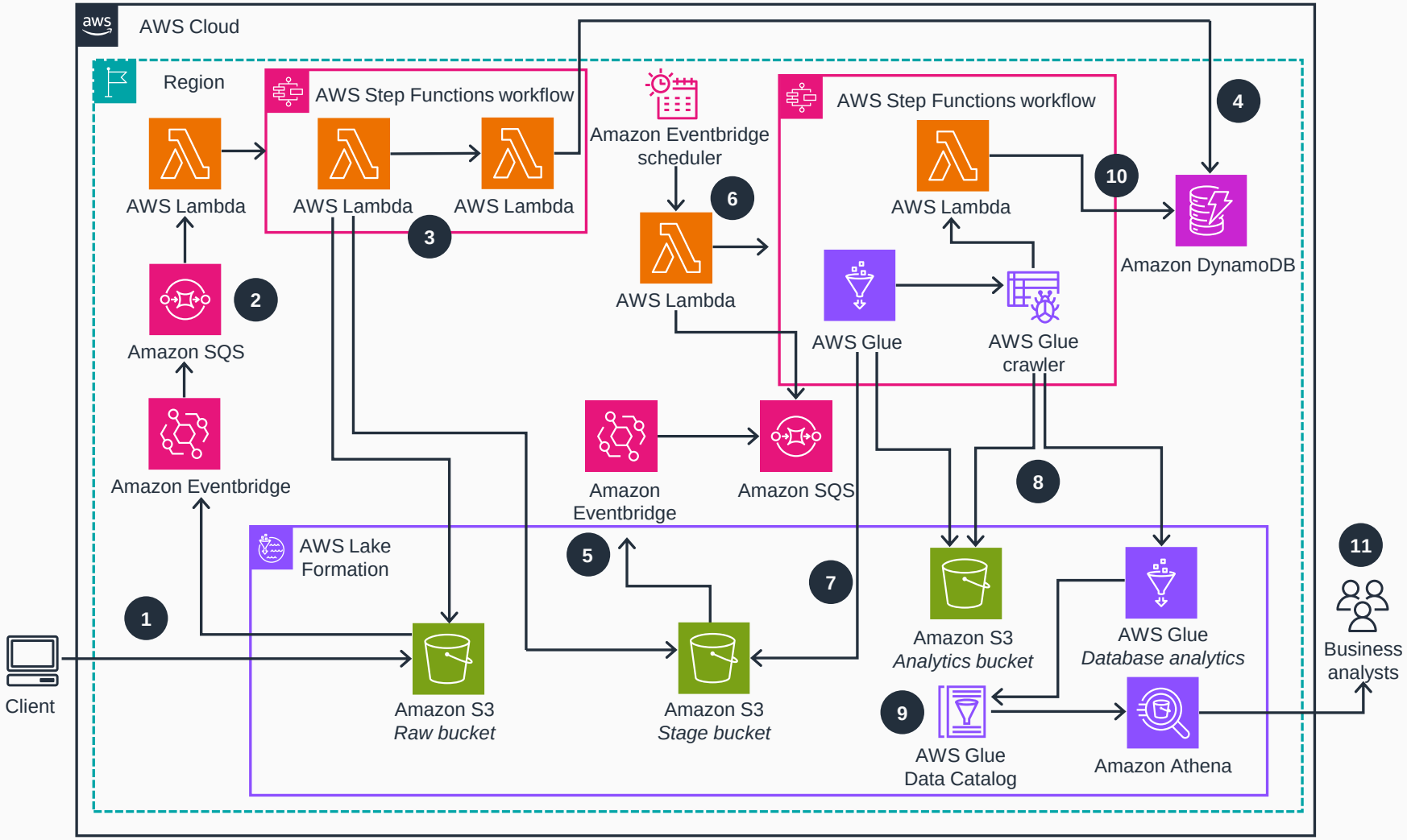
Guidance for Data Lakes on AWS

AWS Serverless Data Lake Framework

This architecture diagram shows how to build a data lake on AWS in addition to demonstrating how to process, store, and consume data using serverless AWS analytics services.



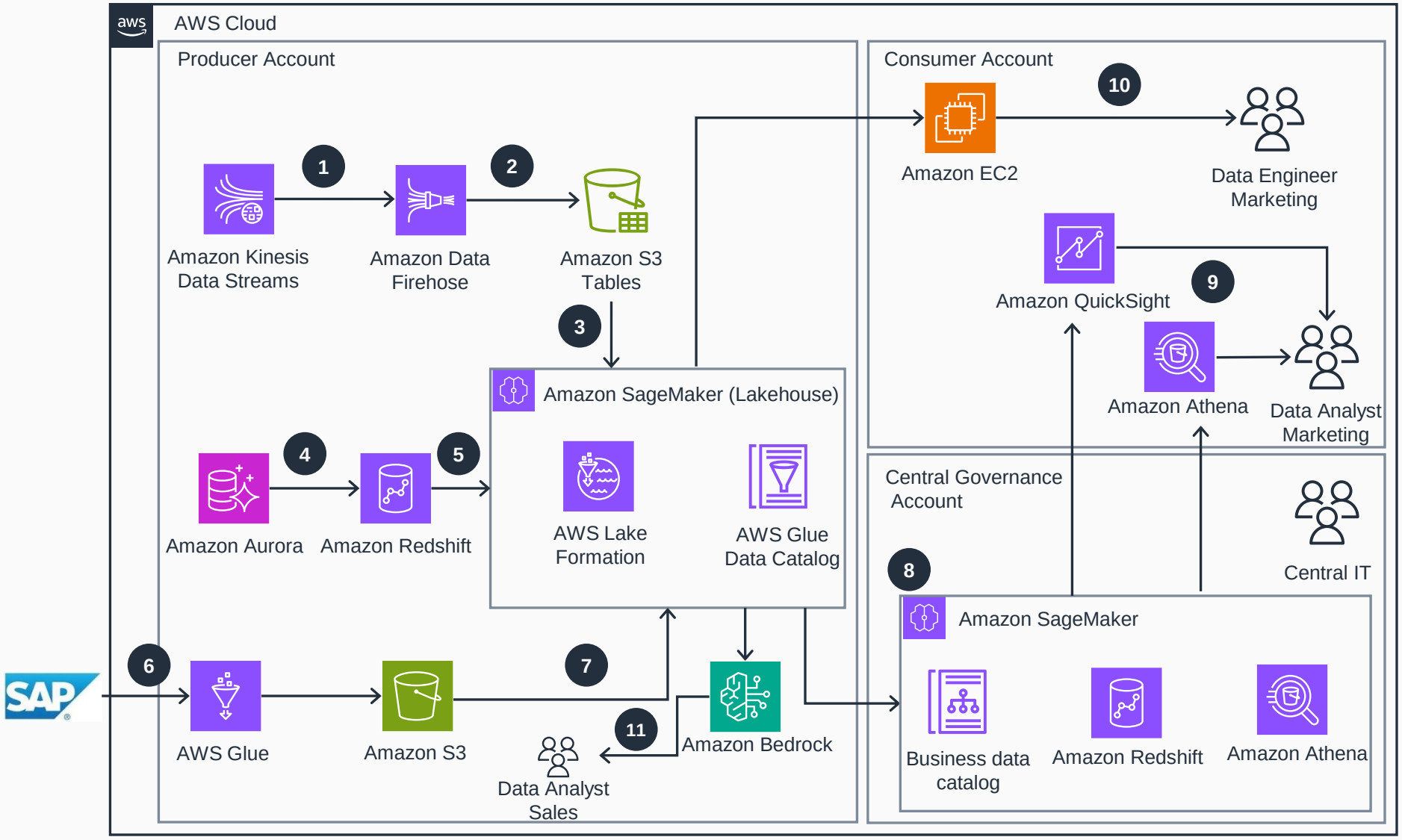
1. The data administrator uploads JSON files in the **Amazon Simple Storage Service (Amazon S3)** raw bucket. Object creation in **Amazon S3** triggers an event in **Amazon EventBridge**.
2. **EventBridge** has a rule that sends a message in **Amazon Simple Queue Service (Amazon SQS)**, which invokes an **AWS Lambda** function.
3. The **Lambda** function triggers the **AWS Step Functions** workflow, in which another **Lambda** function reads files from the **S3** raw bucket and performs transformation. It also writes the new set of JSON files in the **S3** stage bucket.
4. A **Lambda** function updates the **Amazon DynamoDB** table with the **Step Functions** job status.
5. Once the files are created in the **S3** stage bucket, it triggers an event in **EventBridge**, which has a rule that sends a message in **Amazon SQS** with created file details.
6. The **Eventbridge** scheduler runs at certain intervals and invokes a **Lambda** function that retrieves messages from **Amazon SQS** and starts another **Step Functions** workflow.
7. **AWS Glue** extract, transform, load (ETL) reads the data from the **AWS Glue** database stage, then converts the files from JSON to Parquet format.
8. **AWS Glue** ETL writes the Parquet files in the **S3** analytics bucket. **AWS Glue** crawler crawls the Parquet files in the same bucket and then creates analytics tables in **AWS Glue** database analytics.
9. All the staging and analytics catalogs are maintained in the **AWS Glue** Data Catalog.
10. A **Lambda** function updates the **DynamoDB** table with the **Step Functions** job status.
11. Business analysts use **Amazon Athena** to query the **AWS Glue** database analytics.

AWS Reference Architecture

Guidance for Data Lake on AWS

Multi-Source Analytics Lakehouse with AI-Powered Insights

This architecture diagram shows how to build a data lake on AWS in addition to demonstrating how to process, store, and consume data using Lakehouse for Amazon SageMaker and Amazon SageMaker Unified Studio.



1. Ingest store inventory data with **Amazon Kinesis Data Streams,** which feeds the data into **Amazon Data Firehose.**
2. Upload store inventory streaming data into **Amazon Simple Storage Service (Amazon S3)** Tables.
3. Catalog store inventory data into Lakehouse for **Amazon SageMaker**, managed with **AWS Lake Formation**, as a federated **AWS Glue** catalog.
4. Ingest store, product, and promotions dimension data from **Amazon Aurora** (MySQL) to **Amazon Redshift Serverless** via Zero-ETL.
5. Catalog dimension data into Lakehouse for **Amazon SageMaker** as a federated catalog.
6. Ingest store sales data from SAP using **AWS Glue** via Zero-ETL. **AWS Glue** writes the store sales data into **Amazon S3** in Apache Iceberg open table format.
7. Catalog store sales data into Lakehouse for **Amazon SageMaker** as a federated catalog.
8. Control access and governance through **Amazon SageMaker** Unified Studio from the central governance account. The producer account publishes the sales data. The consumer account subscribes and accesses the sales data.
9. The marketing team generates insights from unified data using **Amazon Athena**. The data is pulled from **Lakehouse** for **Amazon SageMaker**. The sales team can also use **Amazon QuickSight** to visualize the data.
10. The marketing team's data engineer with an existing Spark platform accesses sales data from Lakehouse for **Amazon SageMaker** by running Spark jobs on **Amazon Elastic Compute Cloud (Amazon EC2)** using an open Iceberg REST API.
11. Sales team generates insights from unified data using **Amazon Bedrock** foundation models with **Amazon Bedrock Knowledge Bases** using Retrieval Augmented Generation (RAG) in the Producer account by using natural language queries.

Reviewed for technical accuracy October 3, 2025

AWS Reference Architecture