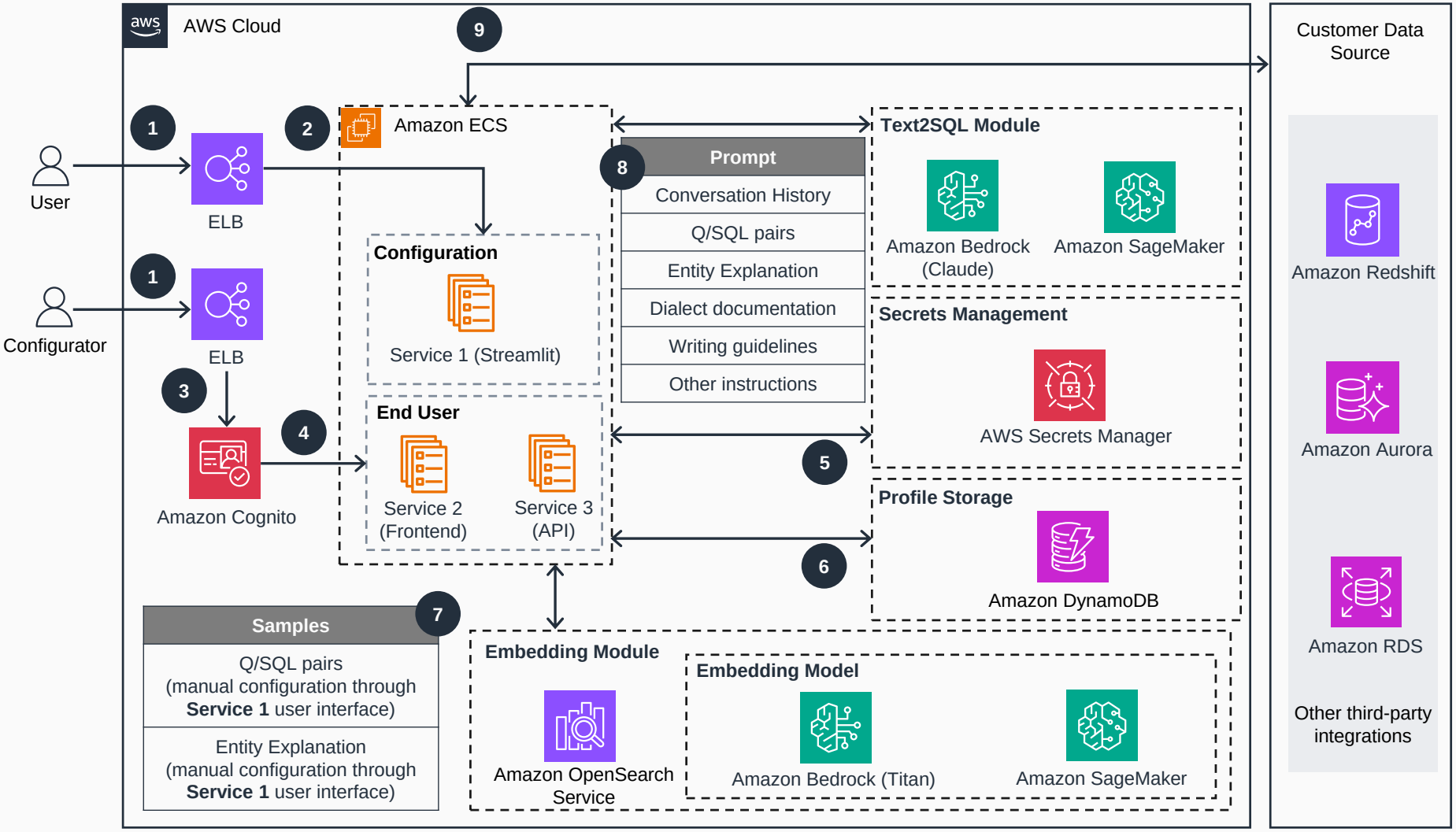
Guidance for Retail Analytics using Generative AI on AWS

This architecture diagram demonstrates how you can use Amazon Bedrock and Amazon SageMaker to build a SQL generator using natural language.



1. The user interacts with the system through **Elastic Load Balancing (ELB)**, which directs traffic to the appropriate service within the AWS Cloud.
2. The configurator uses **ELB** to access configuration-related services.
3. **Amazon Cognito** handles user authentication and authorization, ensuring secure access to the services.
4. The authenticated user accesses the frontend (Service 2) and API (Service 3) hosted on **Amazon Elastic Container Service (Amazon ECS)**, which manages the deployment and scaling of these services.
5. **AWS Secrets Manager** securely stores and retrieves sensitive information, such as database credentials, used by the services.
6. **Amazon DynamoDB** stores user profiles and related data, providing a scalable and high-performance NoSQL database.
7. The embedding module leverages **Amazon OpenSearch Service** and embedding models from **Amazon Bedrock** (Titan) or **Amazon SageMaker** (for example, BGE) to process and index data for efficient querying.
8. LLMs hosted on **Amazon Bedrock** (Claude) or **SageMaker** (for example, Llama 3) converts natural language text into SQL queries, enabling users to interact with databases using plain language.
9. The system pulls data definition language (DDL) information and queries customer data sources such as **Amazon Aurora**, **Amazon Relational Database Service (Amazon RDS)**, **Amazon Athena**, and other third-party integrations to fetch data as required by the user queries.

Reviewed for technical accuracy August 28, 2024

AWS Reference Architecture