System Design Technical Explanation

Introduction

In this design, I have assumed that the data sources, are equipped with mechanisms to send data when available. For this use case purpose, I opted to implement a push ingestion rather than a pull one. By using a *push ingestion*, the system can respond dynamically to incoming data, ensuring timely processing and analysis, which is essential to decrease the time-to-market.

Architecture

Ingestion Layer: Sources data is ingested into a "*Landing Bucket*" from various sources, processed by a Lambda function that manages decryption and/or decompression before moving it to a "*Raw Bucket*" organizing data in partitions. Additionally, data is stored in an S3 *Glacier bucket*.

Normalization Layer: Data normalization occurs through *AWS Glue*, which cleans and parses the data, organizing it into a well-defined format and saving it in the "*Processed Bucket*" in *Parquet* format. A *Glue Crawler* generates the data schema, making it accessible for *AWS Athena*.

Modeling Layer: Once data is saved in the *Processed Bucket*, a Lambda function called "*Model selector*", is triggered to choose the correct data processing *Glue Job*. Normalized data is aggregated and analyzed to generate KPIs based on requirements. Glue saves final data in the "*Exposition Bucket*" and in *RDS* tables for external access.

Data Exposition: Data is accessible through *AWS API Gateway* and can be queried through *AWS Athena* or *RDS*, enabling users to obtain real-time food security alerts.

AWS Services Used

- **AWS Lambda**: Executes serverless functions that ensure high reliability and auto-scaling. The following Lambda functions are used throughout the entire process:
    - decrypting and/or decompressing input data
    - initiating the normalization process
    - starting the modeling process
    - updating Glue Data Catalog metadata (if needed)
- **S3**: Serves as the storage solution for raw and normalized data in various buckets. It provides event triggers to manage data flow efficiently. The S3 buckets used in the process include:

    - Landing, Row, Processed and Exposition Bucket
    - Glacier Bucket
        - Used for long-term data archiving, for data that is accessed infrequently.
- **AWS Glue**: Facilitates data normalization and catalog creation leveraging Apache Spark. The Glue instances used in the process include:
    - Glue normalizing and Glue modeling process
- **RDS**: Stores cleaned relational data.
- **API Gateway**: Exposes APIs to end-users.
- **AWS Athena**: Allows queries on stored data using Glue Data Catalog metadata.
- **Glue Crawler**: Discovers and catalogs data, creating a schema for the datasets.

Data Processing

This event-driven architecture retrieves data from various sources through a *push ingestion mechanism*, where specific events trigger AWS Lambda functions and Glue Jobs to process the data. Given the requirement for near real-time processing, I considered three options

1. a queue system
2. an event-driven system
3. a micro-batching system

Considering the nature of the sources and assuming a low update frequency, I opted for the *event-driven architecture*, which enables rapid data exposure without resource wastage.

Security Considerations

Security measures include controlled access to S3 buckets through bucket policies, Security Groups, VPCs, IAM policies. To secure APIs, an access token can be used.

API Design

REST API design has been implemented to simplify data access for end-users and enhance data consumption. In this context, I will expose a POST endpoint that allows users to retrieve data, specifying range of dates and/or country/region.

Deployment Strategy

CI/CD pipelines and AWS Infrastructure as Code are used to ensure rapid secure and scalable deployment. The pipelines include automated testing and version control. IaC ensures consistent and repeatable provisioning of AWS resources. Monitoring and rollback procedures are integrated to manage any post-deployment issues.