CoreHealthCare

Data Pipeline Design

Presenter: Oyekanmi Lekan

CoreHealthCare

1. Data Ingestion Strategy
2. Data Processing and Transformation
3. Data Storage and Management
4. Pipeline Orchestration and Monitoring

Outline

This capstone project is designed to reinforce thr understanding of the core components of an end-to-end data pipeline. The focus is on conceptualizing and designing a comprehensive data pipeline that addresses data ingestion, processing, storage, and analysis. The project challenges apply theoretical knowledge to a practical scenario without the need to build or code the actual pipeline.

CoreHealthCare

Project Overview

Problem Statement

Highlight CoreHealthCare's challenges with data management (increasing volume, siloed data, inefficiencies in analysis).

Project Goal

Building a robust, conceptualizing and designing a comprehensive data pipeline that addresses these challenges.

Curresnt State

CoreHealthCare collects data from multiple sources, such as EHRs, IoT devices, and third-party APIs, but this data is stored in siloed databases, leading to inefficiencies in access and analysis. The organization struggles with real-time data processing and integrating diverse data formats and is considering cloud-based solutions to improve scalability and performance.

Data is currently stored in siloed databases, leading to inefficiencies in data access and analysis.

The organization faces challenges in processing data in real-time and integrating diverse data formats.

CoreHealthCare collects data from various sources, including electronic health records (EHRs), IoT devices (wearables, medical equipment), and third-party APIs (insurance, public health databases).

CoreHealthCare is exploring cloud-based solutions to enhance scalability and performance.

Task 1: Data Ingestion Strategy

CoreHealthCare collects data from various sources, which can be classified into two main categories:

Structured Data

- **Electronic Health Records (EHRs)**: Contain patient demographics, medical history, diagnoses, and treatment plans.
- **Third-party APIs (insurance, public health):** Provide data on insurance coverage, public health statistics, and other relevant information.

Unstructured Data

- **IoT Devices (wearables, medical equipment)**: Generate time-series data on patient vital signs, activity levels, and device usage.

Task 1: Data Ingestion Strategy

CoreHealthCare collects data from various sources, which can be classified into two main categories:

Batch Processing:

- Suitable for structured data from EHRs and third-party APIs.
- Involves periodically extracting data from source systems and loading it into a data lake or warehouse.
- Tools like Apache Sqoop can be used to transfer data from relational databases to Hadoop or cloud storage.

Micro-batching/Streaming:

- Suitable for semi-structured data from IoT devices.
- Involves processing data in small batches or continuously as it arrives.
- Tools like Apache Kafka can be used to collect and distribute data streams, and Apache Flink can be used for real-time processing.

Tools and Technologies

- Apache Airflow: An open-source workflow orchestration platform that can be used to schedule and monitor data ingestion pipelines.
- Apache Kafka: A distributed streaming platform that can be used to collect, process, and distribute data streams.

Ingestion Method

Data Quality and Reliability:

- **Data Validation**: Ensure that incoming data conforms to expected formats, data types, and business rules. This can involve checking for missing values, invalid characters, and inconsistencies.
- **Deduplication**: Prevent duplicate data from being ingested by using unique identifiers or other techniques to identify and remove duplicate records.
- **Error Handling**: Implement mechanisms to handle errors that may occur during data ingestion, such as network failures, data format issues, or access control problems. This can involve retrying failed operations, logging errors, and notifying administrators.
- **Version Control**: Track changes to data over time to ensure data lineage and accountability. This can be achieved by using tools like Apache Avro to serialize data with schemas.

Task 2: Data Processing and Transformation

The data processing pipeline for CoreHealthCare will involve several key stages:

Data Cleaning

- **Handling Missing Values**: Identify and address missing data using techniques like imputation (e.g., mean, median, mode, or interpolation) or deletion.
- **Standardization**: Convert data to a consistent format (e.g., date, numeric) and ensure uniformity across different sources.
- **Outlier Detection and Correction**: Identify and correct outliers using statistical methods or domain knowledge.
- **Data Type Conversion**: Convert data to appropriate data types (e.g., numeric, categorical) for analysis.

Data Transformation

- **Aggregation**: Combine multiple rows of data into a single row (e.g., calculating averages, sums, counts).
- **Joining**: Combine data from multiple sources based on common keys (e.g., joining patient demographics with medical records).
- **Feature Engineering:** Create new features from existing data to improve model performance (e.g., calculating patient age from birth date).

Task 2: Data Processing and Transformation

Processing Frameworks

- **Apache Spark**: A distributed computing framework that is well-suited for batch processing tasks. It offers a high-level API and can handle large datasets efficiently. Spark can be used for data cleaning, transformation, and aggregation tasks.
- **Azure Databrick**: The Azure Databricks workspace provides a unified interface and tools for most data tasks, including: Data processing scheduling and management, in particular ETL. Generating dashboards and visualizations. Managing security, governance, high availability, and disaster recovery

Data Quality

- **Data Validation Rules**: Define rules to check the validity and consistency of data, such as checking for valid date formats, numeric ranges, and consistency with reference data.
- **Quality Metrics**: Track key data quality metrics, including completeness, accuracy, consistency, and timeliness.
- **Alerts**: Set up alerts to notify data engineers or analysts when data quality issues are detected. This can help identify and resolve problems promptly.

Scalability

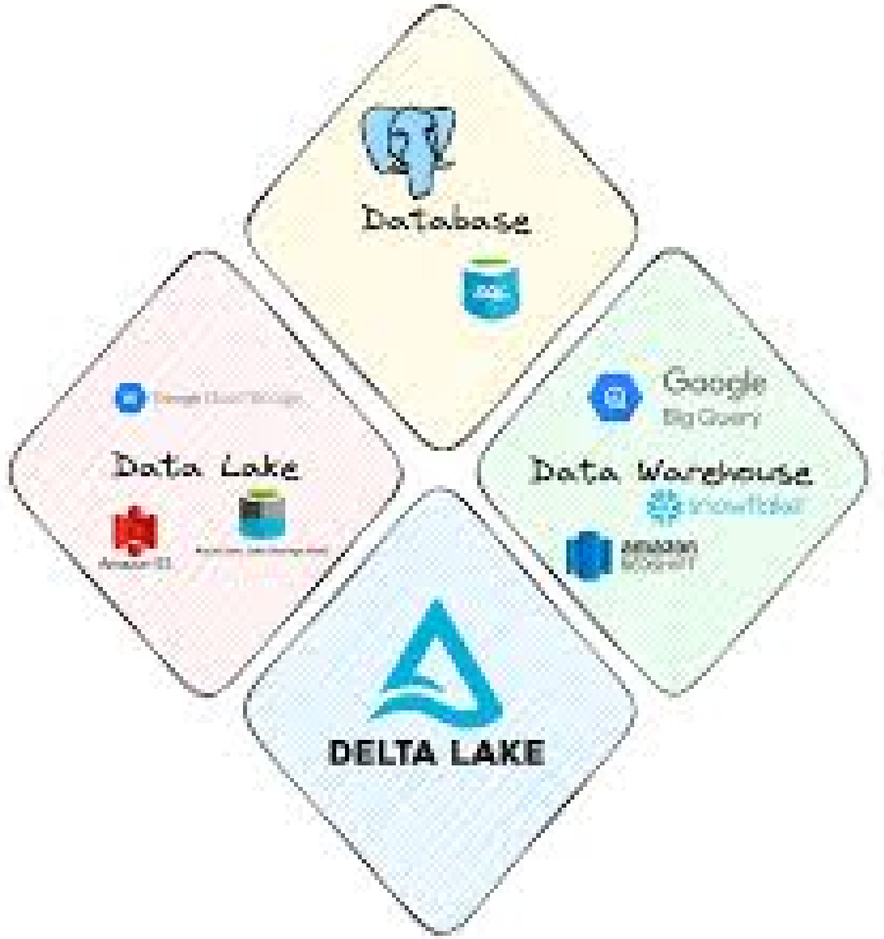
- **Cloud-Based Tools**: Leverage cloud-based platforms like AWS, Azure, or GCP to scale data processing resources dynamically based on workload.
- **Distributed Processing**: Utilize distributed processing frameworks like Spark and Flink to distribute data processing tasks across multiple nodes, improving performance and scalability.
- **Partitioning and Caching**: Partition data into smaller subsets and cache frequently accessed data to improve query performance.

Task 3: Data Processing and Transformation

Design the data storage architecture that supports efficient querying, analysis, and long-term data management.

Storage Solutions

- **Data Lake (Amazon S3)**: A centralized repository for storing large volumes of raw data in its native format. S3 is highly scalable and cost-effective, making it suitable for storing diverse data types.
- **Azure Data Warehouse:** Azure SQL Data Warehouse is designed for enterprise-level data warehouse implementations, and stores large amounts of data (up to Petabytes) in Microsoft Azure.
- **Operational Database (Amazon RDS):** A relational database service that provides a managed database environment for operational applications. RDS supports various database engines, including MySQL, PostgreSQL, and Oracle.



Task 3: Data Storage and Management

Storage Optimizations

- **Data Lifecycle Management:** Implement policies for data retention, archiving, and deletion based on data usage, regulatory requirements, and business needs.
- **Partitioning**: Divide data into smaller, more manageable partitions based on specific criteria (e.g., date, region, customer ID) to improve query performance and reduce storage costs.
- **Denormalization**: Introduce redundancy into the data model by duplicating data to improve query performance. This can be helpful for frequently accessed data that is often joined with other tables.

Security Control

- **Role-Based Access Control (RBAC)**: Assign different roles to users based on their job functions and responsibilities. This allows for granular control over data access and prevents unauthorized access.
- **Encryption**: Encrypt data at rest and in transit to protect it from unauthorized access and data breaches.
- **Audit Logging**: Enable audit logging to track user activity and identify potential security incidents. This can help with compliance and forensic investigations.

Task 4: Pipeline Orchestration and Monitoring

Design the orchestration and monitoring framework for managing and maintaining the data pipeline.

Orchestration

- Apache Airflow: A powerful and flexible open-source workflow orchestration platform that can be used to schedule, monitor, and trigger data pipeline tasks. Airflow allows you to define complex workflows using Directed Acyclic Graphs (DAGs), which represent the dependencies between different tasks.
- DAGs: DAGs are used to visually represent the flow of data through the pipeline. They can include tasks such as data ingestion, processing, transformation, and storage. Airflow provides a web interface for creating, editing, and monitoring DAGs.
- Scheduling: Airflow allows you to schedule tasks to run at specific times or on a recurring basis. This can be useful for automating routine data processing tasks.

Monitoring

- CloudWatch: A monitoring service provided by AWS that can be used to track the performance of your data pipeline. CloudWatch can monitor various metrics, including CPU utilization, memory usage, network traffic, and error rates.
- Custom Metrics: You can create custom metrics to track specific aspects of your pipeline, such as the number of records processed or the time taken to complete a task.
- Dashboards: CloudWatch allows you to create custom dashboards to visualize key metrics and identify trends over time.

Task 4: Pipeline Orchestration and Monitoring

Design the orchestration and monitoring framework for managing and maintaining the data pipeline.

Alerts and Logging

- Alerts: Set up alerts to notify you when key metrics exceed thresholds or when errors occur. This can help you identify and resolve issues promptly.
- Logging: Implement logging to record information about the execution of your pipeline tasks. This can be helpful for troubleshooting problems and auditing data processing activities.
- Log Aggregation: Use a log aggregation tool like Amazon Kinesis Firehose or AWS CloudWatch Logs to collect and analyze logs from multiple sources.

Scalability

- Auto-scaling: Leverage auto-scaling features provided by cloud platforms to automatically adjust the resources allocated to your pipeline based on workload. This can help ensure that your pipeline can handle varying data volumes and processing loads.
- Elastic Computing: Use elastic computing services like AWS EC2 to dynamically scale the number of instances used by your pipeline.
- Serverless Computing: Consider using serverless computing services like AWS Lambda to execute individual tasks within your pipeline without having to manage servers.

Task 3: Data Storage and Management

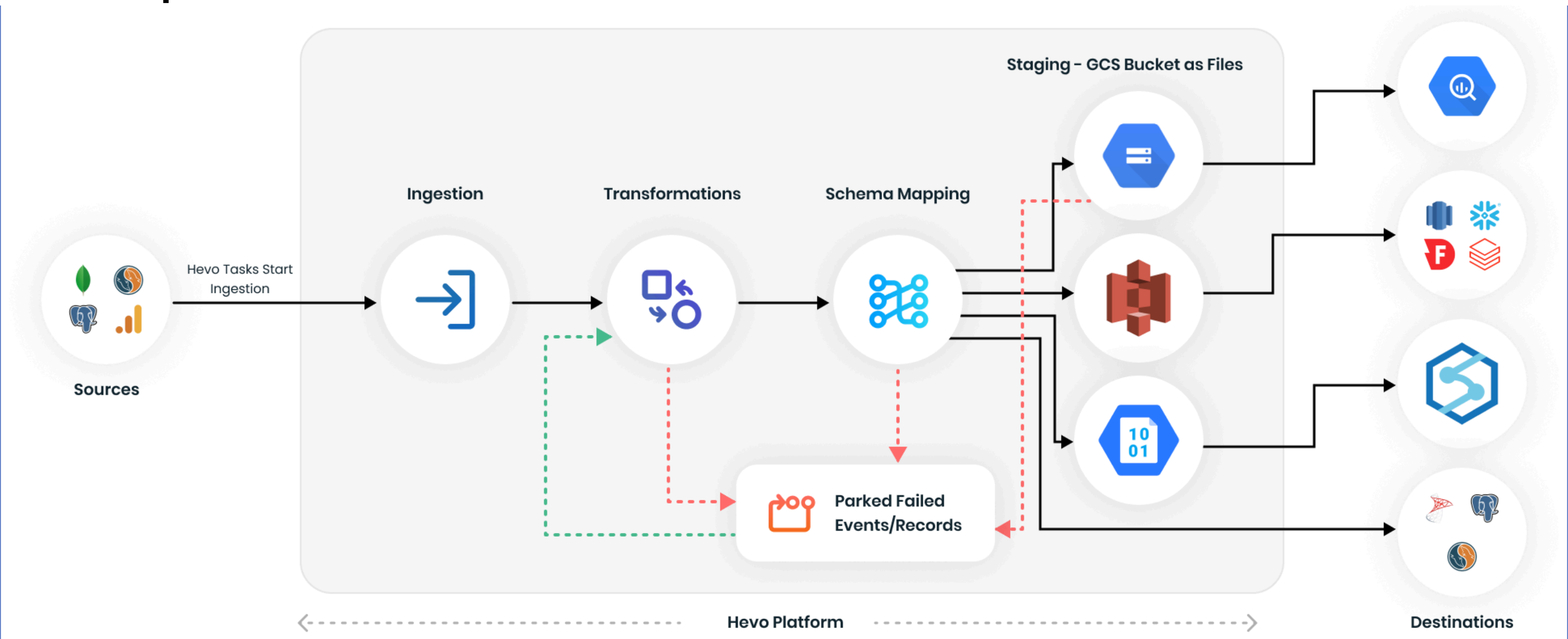
Storage Optimizations

- **Data Lifecycle Management:** Implement policies for data retention, archiving, and deletion based on data usage, regulatory requirements, and business needs.
- **Partitioning**: Divide data into smaller, more manageable partitions based on specific criteria (e.g., date, region, customer ID) to improve query performance and reduce storage costs.
- **Denormalization**: Introduce redundancy into the data model by duplicating data to improve query performance. This can be helpful for frequently accessed data that is often joined with other tables.

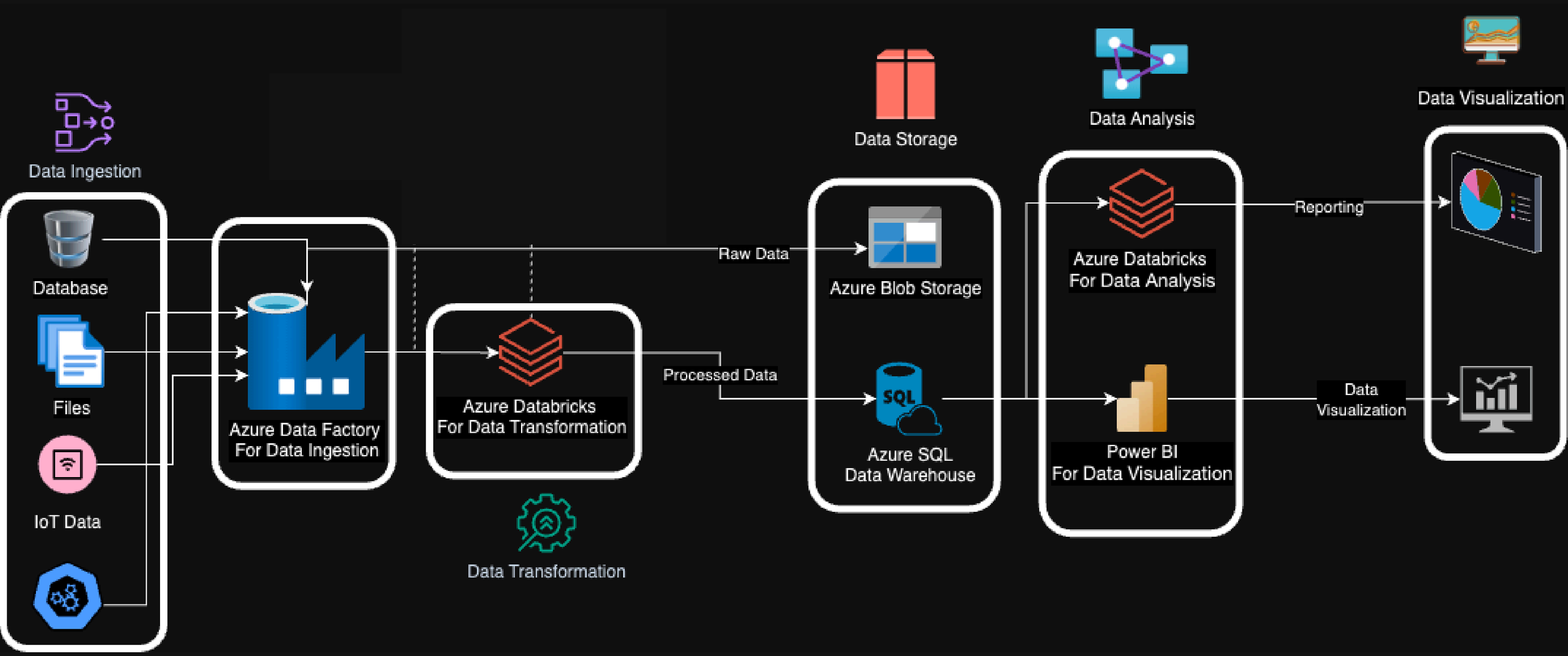
Security Control

- **Role-Based Access Control (RBAC)**: Assign different roles to users based on their job functions and responsibilities. This allows for granular control over data access and prevents unauthorized access.
- **Encryption**: Encrypt data at rest and in transit to protect it from unauthorized access and data breaches.
- **Audit Logging**: Enable audit logging to track user activity and identify potential security incidents. This can help with compliance and forensic investigations.

ETL Pipeline



CoreHealthCare’s End-to-End Data Pipeline Architecture



Thank
You