Data Lake Architecture -

A Comprehensive Design Document

Medical Data Processing Company

# Tracker

## Revision, Sign off Sheet and Key Contacts

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 04/10/2024 | Alexander Milosz | 1 | First Submission |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  | Udacity Reviewer |  |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
|  | Data Architect | Medical Data Processing |  |

# Purpose

The document presents a detailed Data Lake Architecture Design for a Medical Data Processing Company, which encounters several problems with their current solution and cannot meet their needs and requirements. Therefore, a new architecture is proposed, which ensures scalability, reliability and compliance while supporting real-time and batch data processing for multiple use cases like machine learning, ad-hoc queries and reporting. The target audience is technical people, such as enterprise architects, software engineers and technical directors.

In-scope:

Data Ingestion, Storage, Processing, Serving Layer

Out-of-scope:

Vendor-locked solutions, Data warehouse design, Data Quality and Monitoring

# Requirements

The Medical Data Processing Company faces challenges with their current SQL Server-based architecture, which struggles to scale, resulting in slow ETL, queries and even system crashes. The company needs a new architecture to handle increasing data volume (growing each year), enable near-real-time analytics and support machine learning and ad-hoc querying.

The existing technical environment consist of:

* FTP servers and API extract agents handling data ingestion
* SQL Servers (Master and Stage) running ETL jobs
* Web and Application servers for customers

The current data volume is:

* 8,000 + facilities
* Handling compressed ZIP files in different size and different record file types (such as CSV, TXT and XML)
* Handling 15 million files a day
* Data growth rate of 15-20% year over year.

Business related and technical requirements have been derived from the Company Profile Problem Statement:

Business Requirements:

* Improve system uptime to ensure reliability
* Reduce query latency enabling faster queries and report generation
* Scalability – ensure that the architecture scales with increasing data volume and velocity
* Embrace open-source tools to avoid vendor lock-in
* Metadata-driven design to use common frameworks to minimize custom developments
* Centralize data storage for easy access and avoiding data silos

Technical Requirements:

* near-real-time processing instead of nightly batch loads
* separation of metadata, data and processing layer
* unlimited historical data
* scalable processing speeds
* fault tolerant
* perform CDC (Change Data Capture)
* ML, visualization and reporting without moving data
* Ad-hoc queries support interactive querying and analytics

# Data Lake Architecture design principles

1. Scalability:  
   The architecture needs to scale horizontally as data volume and velocity grows. This principle is crucial for a growing company. It ensures that the system remains efficient and performant.
2. Separation of Concerns  
   The design must enforce a clear separation between metadata, data and processing layers. It ensures that each layer can be managed and scaled independently. This principle ensures that the architecture becomes more modular and easier to manage, allowing being scalable and flexible.
3. Open-Source:   
   Using such tools ensures the architecture to vendor lock-in and is flexible enough to adapt to new frameworks and technologies as needed. This principle ensures that the architecture can evolve over time without being constrained by proprietary tools.

# Assumptions

1. Data Volume Growth: Using Amazon S3 as a scalable storage, the architecture can scale without performance degradation.
2. Data Structure is mainly consistent: It is assumed that the incoming data remains mainly consistent in terms of schema and structure. This consistency is necessary for a metadata-driven design. Otherwise, additional data transformation logic may be needed, which will introduce more complexity.
3. Real-Time Processing is not critical: The architecture is not designed to stream data or process data in exact real-time. Otherwise, additional tools such as Apache Kafa may be needed to stream data directly from the Ingestion layer into the Serving layer.

Potential risks in the future:

* Data Quality Issues: If data arrives inconsistent or corrupted the data lake may accumulate poor-quality data, which can affect downstream processes like ML. This could be solved by implementing data quality checks during ingestion and processing.
* Vendor Dependency on S3: Although the architecture avoids vendor lock-in the storage layer consist of S3 buckets with managed IAM roles and AWS Backup plans. It may be considered to approach a multi-cloud-strategy.

# Data Lake Architecture for Medical Data Processing Company

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte Beschreibung

# Design Considerations and Rationale

## Ingestion Layer

The company ingests data from medical facilities in the form of ZIP files, which contain XML, CSV and TXT records. The data is compressed, and password protected. Data is ingested via FTP servers (pushed) or pulled through APIs using API agents. The ingestion happens on the fly (near-real-time).

Ingestion from Databases, FTP Servers and APIs:

Databases: SQL queries are run at very frequent intervals. CDC pipelines are implemented for RDBMS over Apache Nifi (see link below).

FTP Servers: Push data from Medical Facilities to FTP Servers in real-time for new incoming files. Files can be decompressed and decrypted for further processing.

API: Pull data from APIs at Medical Facilities with API agents (configured in Apache Nifi). Process is configured as triggered based.

As mentioned above Apache Nifi will be used due to its flexibility when it comes to handling diverse data formats and pipelines. Data can be decompressed and decrypted after ingestion. Nifi can connect to Databases, can monitor files from FTP Servers and can pull data from APIs.

Furthermore, Nifi has a cluster mode, which allows to add more nodes (horizontally scaling). It can handle batch and stream processes in near-real-time.

AWS Glue and Azure Data Factory were rejected due to vendor lock-in. Kafka was considered for real-time data ingestion but not used as near-real-time processing is sufficient.

## Storage Layer

The system uses Amazon S3 as a scalable, cost-effective solution for storing raw, transformed, and snapshot data.

S3 is fully managed and scales automatically. Older data can be moved to colder storage like S3 Glacier (Lifecycle Policies).

For legal and regulatory reasons, data needs to be versioned and backed-up frequently (e.g. nightly jobs). Amazon S3 supports Versioning, Cross-Region Replication and even Object Lock for legal and compliance reasons (write once and read many).

Furthermore, AWS Backup can automate backups and simplify recovery to ensure a reliable, fault-tolerant system. Backups of HBase and Solr are done by frequent snapshots and exporting them into an S3 bucket. Both are replicated along various nodes.

Apache Atlas will be used for metadata management and tracking data lineage. Technical data, like information about file formats, sizes, schemas and transformation processes will be stored. Business metadata, like tags or descriptions (e.g. patient data, health metrics, compliance information) will be stored.  
Tracking the Lineage, will ensure to keep track how the data moves through the whole pipeline, enabling data governance and auditing. Custom data can be applied to the objects, such like attributes (e.g. Facility name, timestamp…).

To ensure highly compressed, cost-effective and optimized for querying data Parquet and Avro formats will be used. Parquet is in columnar format and therefore ideal for analytics. Avro on the other hand is in row-based format and supports schema evolution.

Sensitive medial data needs to be secured efficiently. Data needs to be encrypted at rest and in transit to ensure legal regulations. To do so, SSE-KMS (see link below) in S3 can encrypt data with encryption keys. In Transit data can be encrypted using SSL/TLS.

Role-Based Access Control should be considered with AWS IAM Roles to control data access. For fine-grained data access control across the data pipeline Apache Ranger can be implemented with Apache Atlas to ensure a metadata-based policy.  
  
Furthermore auditing and monitoring needs to be done. For the cloud AWS CloudTrail can be used. For the pipeline Apache Atlas can be used to audit.

Other Tools Considered: HDFS was rejected as an on-premises option due to high operational overhead compared to S3.

## Processing Layer

Data is processed with Apache Spark for batch workloads, transforming raw data into structured formats in S3.

Batch: For batch workloads, Apache Spark would perform bulk transformations if data stored in S3 raw data bucket.

Realtime: While true real-time processing isn’t required, Apache Nifi will handle near-real-time ingestion by pulling data as soon as it's available and storing it in S3.

CDC: Nifi tracks changes in SQL Server databases for continuous data ingestion.

Presto is used for SQL-based querying of S3-stored data without moving it to a traditional database.

Used Tools:  
Apache Nifi for data ingestion

Apache Spark for processing

Presto for querying

Apache Atlas for metadata

Apache Ranger for security

Apache Airflow orchestrates tasks and schedules jobs

AWS Glue and Azure Synapse were not selected to avoid vendor lock-in.

The architecture scales horizontally by adding more nodes to handle large datasets efficiently.

## Serving Layer

The serving layer is where the transformed data (cleaned, aggregated, and formatted) is stored and made accessible for downstream applications.

The serving layer is critical for fast querying and efficient access to data.

Data in the serving layer is in the S3 Transformed Bucket, ready for consumption by machine learning models, dashboards, and reporting systems.

The serving layer supports ML models (e.g., TensorFlow), dashboards (PowerBI, Tableau, MicroStrategy), and the generation of daily, weekly, or nightly reports using SQL or ad-hoc queries with Presto.

# 7. Evaluation of different storage and processing frameworks

As the company continues to grow, the architecture will need to evolve to address new challenges and opportunities. Including Apache Kafka to enable real-time-processing and streaming of data for example, could be a future development.

The next step for the design and data team could be:

* establish a Data Governance Framework to define roles and responsibilities for data stewards, data owners.
* develop a centralized data catalog.
* Implement continuous Data Quality checks.

Moving forward, close collaboration between the architecture and data teams will be key to refining the system, ensuring it scales smoothly and supports various needs.

# 8. Conclusion

This document outlines a comprehensive Data Lake Architecture for the Medical Data Processing Company, addressing the key challenges and keeping track of the business and technical requirements. The proposed design leverages open-source tools combined with the scalability and flexibility of Amazon S3 cloud storage. The architecture ensures near-real-time data processing and supports multiple use cases.

# 9. References

Ingestion:

<https://nifi.apache.org/documentation/v2/>

<https://hevodata.com/learn/apache-nifi-data-ingestion/>

<https://pratikbarjatya.medium.com/building-data-ingestion-system-using-apache-nifi-76e90765ac43>

Storage:

<https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/amazon-s3-data-lake-storage-platform.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/replication.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html>

<https://docs.aws.amazon.com/security-lake/latest/userguide/data-protection.html>

<https://atlas.apache.org/#/Architecture>

Processing:

<https://airflow.apache.org/docs/apache-airflow/stable/index.html>

<https://spark.apache.org/docs/latest/>

Serving:

<https://cwiki.apache.org/confluence/display/Hive/>

<https://prestodb.io/docs/current/>

[https://www.projectpro.io/compare/apache-hive-vs-aws-presto#:~:text=Hive%20is%20used%20for%20batch,and%20interactive%20data%20analysis%20tasks.](https://www.projectpro.io/compare/apache-hive-vs-aws-presto%23:~:text=Hive%20is%20used%20for%20batch,and%20interactive%20data%20analysis%20tasks.)