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 |
| 24/03/2023 | Sonia Fogari | 0.1 | Initial draft |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
| FirstName LastName | 1.0 | Udacity Reviewer  Enterprise Data Lake Architect |  |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
| Sonia Fogari | Data Architect | Medical Data Processing | student@email.com |

# Purpose

This document contains the description of the architecture of an IT system which aims at fulfilling the business and technical requirements of the company Medical Data Processing Co. This company is experiencing a 15-20% yearly data growth and the IT solution currently in use for the data storage and processing does not scale up enough to meet the business needs, due to its insufficient performance and frequent business interruptions. The company would like to leverage the advantages of implementing a data lake as a robust and flexible IT solution to support the planned business growth.

In this document we provide details about a suitable IT architecture (including ingestion, storage, processing and serving layers) of a data lake designed to meet the specific requirements of the company and explain why these components outperform the currently implemented IT solution.

Based on this document, the board of the company, supported by the technical expert team, will decide if the proposed data lake design concept should be implemented and allocate the necessary budget and resources for its implementation.

The technical expert team as the main target audience of this document consists of data engineers, software engineers, functional and operations managers, enterprise architects and security managers of the affected IT applications and their managers/directors.

In scope:

* Applied data lake design principles
* Data lake major components
* Rationales for the proposed data lake components

Not in scope:

* Timeline / plan for implementing the proposed data lake
* Economic evaluation of the proposed solution
* Technical specification of each IT component and interface contract
* Data modelling
* Business processes

# Requirements

The current IT architecture solution, which is based on a SQL server, does not scale efficiently to support growing data volumes. This can be observed by frequent service interruptions and an insufficient performance in processing the data. Moreover, to cope with these high processing performance requirements, in the current solution data needs to be duplicated into different systems, causing increasing storage costs and error-prone processes. Another consequence of the restricted processing capacity are limitations in the data amount which can be stored and processed, preventing the possibility to store and analyse historical data in addition to the current data. Storing historical data would be beneficial because it could open the possibility of leveraging new business areas such as machine learning and near-time analytics use cases and generate revenues from them.

The existing technical environment is constituted by the following components:

● one Master SQL DB Server

● one Stage SQL DB Server

● 3 other smaller servers for Data Ingestion (FTP Server, data and API extract agents)

● Series of web and application servers

The medical company processes small data files (99% files size ranges from 20 KB to 1.5 MB, edge cases up to 40MB) from about 8k facilities. The files are in a zip format, and when unzipped will provide either CSV, TXT, XML records. The number of records processes per day is on average 15Mio, with a data volume growth rate of 15-20% yearly.

Business requirements include the improvement uptime of the overall services, the avoidance of vendor lock-in from the adoption of proprietary solutions, the flexibility to expand the services and integrate new frameworks, the compliance to regulations in processing personal data, the control of the maintenance costs by designing general scripts based on the metadata instead of multiple ad-hoc scripts and the ease of access to the system.

Technical requirements include the ability to process the data at the time the files are received, the ability to keep unlimited historical data, the ability of identifying and capturing changes made to data in a database and then delivering those changes in real-time to a downstream process or system and the ability to drive multiple use cases supported by different tools and technologies from the same dataset, without the need to move the data or extract the data.

In particular, the requirement about the compliance to data protection law, which I have seen in some companies operating in the EU markets (where the data protection laws are stricter), can be very hard to fulfil if the architecture of the system is not supporting this by design. According to these regulations, the customers might ask to delete all their personal data at once, and this can be very hard to fulfil if the customer data is spread among more systems. Since the medical company processes personal data, it might be wise to consider the adherence to the data protection requirements from the beginning.

# Data Lake Architecture design principles

The first and most important design principle which is applied is to **get advantage from the implementation of a Big Data system** instead of a traditional data-processing application software to store and process the data. This principle needs to be followed because the existing systems can't scale anymore, and the capabilities have already been maxed. The application of this principle will help meeting all most important business and technical requirements: improved uptime service due to absence of a single point of failure, higher performance of queries, ability to cope with 20% yearly data growth with no significant business interruptions and ability to store a larger amount of data, including historical data.

The second design principle is **the implementation of Big Data Security management best practices** wherever possible, such as Safeguard Distributed Programming Frameworks, Big Data Cryptography and Granular Access Control (reference: <https://www.xenonstack.com/blog/big-data-security>). Big data security ensures a higher availability of the system, because it gets less vulnerable to cyber security attacks and improves the protection of the customer data.

The third design principle is **favouring solutions which ensure a higher flexibility** (reference: <https://towardsdatascience.com/how-to-set-up-an-flexible-and-scalable-data-analytics-platform-quickn-easy-5fb3a4c83745>). This means concretely the adoption of cloud services instead of self-hosted solutions, the storage of the data in the raw format in the staging area (Data lake vs. Data Warehouse – reference: <https://www.interviewbit.com/blog/data-lake-architecture/> ) and in general the adoption of technologies with a high market share. The advantages of implementing a flexible solution are processing and storage costs depending on the actual data volumes, faster implementation time, faster changes and error fixing processes and the ability to derive multiple use cases from the same data.

# Assumptions

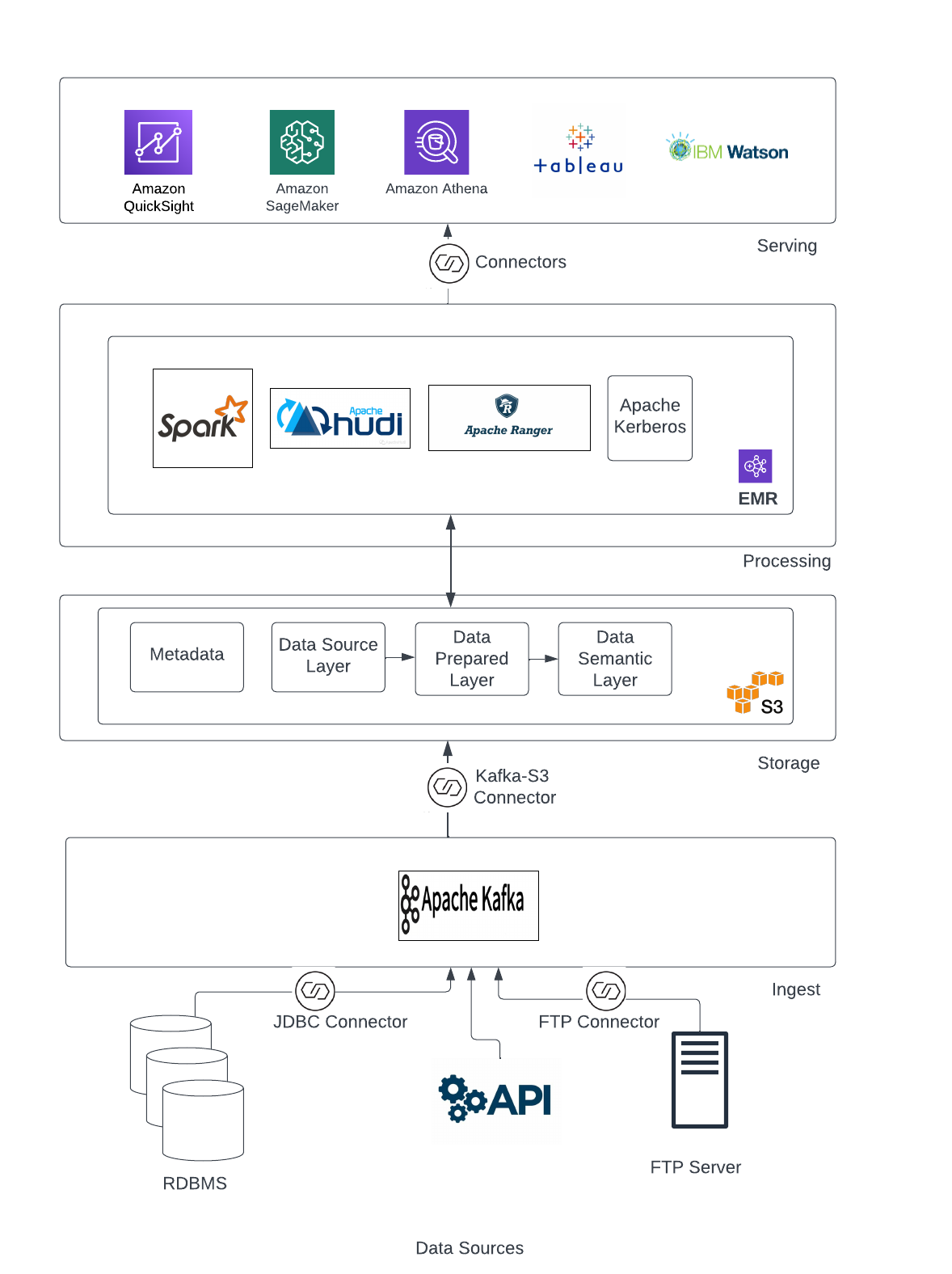
Following assumptions are considered:

1. The medical company has a cloud first strategy, does not consider the option of owning and managing its own hardware as business critical
2. the IT and business teams are willing to accept new challenges and learn something new, and they have the time to participate to trainings, in case they don´t have the technical and organizational skills required by the new solution.
3. the data from the medical facilities contains personal identifiable data such as patient name, insurance number and so on, which need to be de-personalized to be processed safely in the upstreaming systems for analytical purposes.

Potential risks based on the assumptions:

1. Loss of know-how and human resources who currently deal with hardware and system management
2. Costs and time for managing the unexpected turnover of the employees, additional costs for unplanned training and change management measures
3. If data protection management will never be a requirement, or if all the data are already ingested in an anonymized form, then the architecture designed to support compliance and data protection processes is unnecessary complex (over-designed).

# Data Lake Architecture for Medical Data Processing Company



# Design Considerations and Rationale

## Ingestion Layer

In the proposed architecture all types of data (XML, CSV and TXT files .gzipped and protected by password coming from about 8000 medical facilities) are managed by the same data ingestion tool, which is **Apache Kafka**. Kafka is designed to ingest all requested data types (reference: https://www.kai-waehner.de/blog/2020/09/25/kafka-xml-messages-transformation-connector-middleware-comparison-connect-smt-esb-etl-web-services-soap-wsdl-schema/) .

Kafka can also support the ingestion of data from different sources, from a technical point of view (data coming from Databases, FTP Servers and APIs) into the storage level of the Data Lake: in order to implement the connectivity specific connectors must be implemented, such as the JDBC source connector to ingest data from Databases (references: <https://hevodata.com/learn/kafka-connectors/> and <https://acloudguru.com/hands-on-labs/importing-data-from-a-database-with-kafka-connect> ) and the FTP source connector to ingest data from FTP servers (<https://docs.lenses.io/5.0/integrations/connectors/stream-reactor/sources/ftpsourceconnector/>).

Apache Kafka and the required pre-built connectors are open-source tools. Kafka can be used for both analytical and transactional data (reference: <https://www.kai-waehner.de/blog/2022/03/09/analytics-vs-transactions-api-data-streaming-with-apache-kafka/>) in the same tool. The possibility of ingesting analytical data as well, in addition to the transactional data, provides more flexibility to the whole solution, for example in case the medical company decides to integrate in the future additional data types such as results from analyses on the raw medical data from the medical facilities or from other medical data processing providers.

Kafka is designed for the ingestions into data lakes, and it is scalable by design up to up to a thousand brokers, trillions of messages per day, petabytes of data and hundreds of thousands of partitions (reference: <https://kafka.apache.org/> ). In order to scale with the growing data volumes, the medical company would need to change the cluster capacity in Kafka accordingly as a manual process. This process could be avoided by adopting a Kafka solution managed by the could provider, such as Amazon MSK (reference: <https://aws.amazon.com/msk/> )

Due to the fact that 80% of all Fortune 100 companies currently use Kafka (reference: <https://kafka.apache.org/>), this tool seems to be the open-source data ingestion tool with the highest market share.

Other tools considered:

* Amazon MSK: 3rd party tool. Shortcoming: vendor lock-in.
* Apache Storm: open-source tool. Shortcoming: the primary use of Strom is Stream Processing, whereas for Kafka is a Message Broker (reference: <https://www.educba.com/apache-storm-vs-kafka/>) . Stream processing functionalities do not fit well with the main medical company use cases (tool would be over-engineered).
* Apache Flume: open-source tool. Shortcomings: Flume is not scalable in comparison with Kafka (reference: <https://www.geeksforgeeks.org/difference-between-apache-kafka-and-apache-flume/> )
* AWS Kinesis: 3rd party tool. Shortcoming: vendor lock-in, limited market share (reference: <https://www.instaclustr.com/blog/aws-kinesis-vs-kafka-comparing-architectures-features-and-cost/> )

## Storage Layer

In the proposed architecture the data is stored in **Amazon S3 buckets**. The total number of data and the total number of objects which can be stored in S3 buckets is unlimited (reference: <https://aws.amazon.com/s3/faqs/#:~:text=The%20total%20volume%20of%20data,a%20maximum%20of%205%20TB>. ) This allows to store all data from the medical company, including historical data, and manage a 20% YoY Data Growth rate.

Another characteristic of the data managed by the medical company is the small size of the files. This issue needs to be addressed because in general, big data systems are optimized to manage bigger files and the management of smaller files could cause performance issue in the data processing, due to the high proportion of metadata in the buckets in relation to the data contained in the bucket. To address this issue, it is recommended to compact the files to about 100s of MBs of range using managed services, such as via managed Spark Services or Amazon EMR (this is done in the processing layer). Reference: <https://www.upsolver.com/blog/small-file-problem-s3>.

Back-up and recovery can be handled using the AWS Backup service (reference: <https://docs.aws.amazon.com/aws-backup/latest/devguide/restoring-s3.html>). As a prerequisite, the medical company must set up the recovery points and create the backup and recovery plan in advance. The AWS Backup service is an off-the shelf service offered by Amazon on the top of the storage in the s3 buckets which is very useful and low-barrier.

The s3 buckets must store both data from the medical facilities and metadata separately.

Descriptive metadata for electronic medical records typically consists of application metadata (i.e., Patient last name), document metadata (i.e., username who created the document and creation date), file metadata (i.e., name of the file) and embedded metadata (i.e., versioning). Reference: <https://library.ahima.org/doc?oid=106378#.ZBnQR3aZOUl>.

In addition to the descriptive metadata, there are structural metadata, such as the number of pages in a document, preservation metadata, such as retention period, administration metadata, such as security classification of document, provenance metadata, such as medical facility sending the file and definitional metadata, such as vocabulary and business rules to process the data (reference: https://www.spiceworks.com/tech/devops/articles/what-is-metadata/ ) . As a best practice, is recommended to store and manage the metadata in a metadata management system (reference: <https://www.techtarget.com/searchstorage/feature/6-best-practices-for-metadata-storage-and-management>. )

As a format of the data in the semantic layer, we plan to use Parquet. This file format has been chosen because it is open-source, stores the data in columns (this saves storage space compared to the .csv format on Amazon s3, reference: <https://www.databricks.com/glossary/what-is-parquet> ) and supports schema evolution, which is a great feature since the incoming data from 8k medical facilities are very heterogeneous and subject to changes in the schemas. The file conversion from the original file format to Parquet must be managed by a tool, such as Apache Spark or EMR, in the processing layer (reference: <https://sparkbyexamples.com/spark/spark-convert-csv-to-avro-parquet-json/>).

In order to secure the data, fine-grained access control to sensitive data and data encryption are implemented. Data encryption is available by default on s3 buckets, with no additional costs or impact on the performance (reference: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/default-bucket-encryption.html> ).

Fine-grained access control and in general, the management of security and compliance requirements is implemented through 3 different (sub-) layers for data within the data storage layer (reference: <https://lingarogroup.com/blog/data-lake-architecture>) :

* Source layer, where raw data is stored
* Prepared layer (aka. transformation layer), where data is standardized, cleansed, and transformed
* Semantic layer (aka. application data layer), where the data is accessed be the applications and analytics/visualization tools.

The data transformation in the prepared layer includes the depersonalization (and re-personalization of data): if the same dataset needs to be accesses by 2 applications, one using the data in personalized form (for example, to contact the patient) and the second in depersonalized form (where the name of the patient is converted to a unique identifier to be used for aggregation / analytical purposes), then the 2 datasets are stored separately on the semantic layer, with different access control policies.

The fine grained access control in the semantic layer is implemented using Apache Atlas to maintain the classification metadata and visualize the data lineage through the layers (reference: <https://www.datasciencecentral.com/how-to-discover-and-classify-metadata-using-apache-atlas-on/> ). It enables controls on access to entity instances and operations like add/update/to remove classifications. Integration with Apache Ranger enables authorization/data-masking on data access based on classifications associated with entities in Apache Atlas. These 2 tools, both-open source, belong to the processing layer but their usage ensure a centralized security management of the data in the storage layer.

In addition to the data storage solution proposed in this data lake architecture, which is Amazon s3 buckets, other tools have been considered:

* Apache Hadoop: open-source tool. Shortcoming: lower scalability, higher prices, lower performance. References: <https://www.integrate.io/blog/storing-apache-hadoop-data-cloud-hdfs-vs-s3/> and <https://www.oursecondinnings.org/post/why-is-hadoop-dead-is-big-data-dead>.
* HDFS managed by Cloudera: 3rd party tool. Shortcoming: higher prices. Reference: <https://www.websitebuilderinsider.com/what-is-the-difference-between-cloudera-and-aws/>
* Other cloud storage 3rd party vendors, such as Google Cloud, Microsoft Azure. Shortcoming: lower market share.

## Processing Layer

The main process activities on the data for electronic medical records include moving, cleaning, splitting, merging, sorting and validating (Reference: <https://www.researchgate.net/figure/Optimizing-the-Electronic-Health-Records-through-Big-Data-Analytics_fig1_335633872>).

Due to the architecture of the storage layer, data and metadata must be stored and managed separately. Due to the data protection and security requirements, data needs to be moved from the source layer to the preparation layer, to the semantic layer. Data needs to be converted to the target file format on the semantic layer, which is Parquet. Data needs to be de-personalized when needed. Data needs to be deleted from all applications / storage when requested by the customer.

In order to satisfy the different processing needs (batch and real-time), the most widely used and scalable open-source tool is **Apache Spark** (reference: <https://spark.apache.org/> ). Apache Spark supports all these processing needs natively, as well as the possibility to implement ad-hoc queries in SQL in one tool.

The integration of Spark with the Amazon s3 buckets is provided by the **Amazon EMR API** (reference: <https://aws.amazon.com/emr/features/spark/#:~:text=You%20can%20quickly%20and%20easily,or%20the%20Amazon%20EMR%20API>.)

To fulfill additional processing requirements which are not supported natively by Spark, additional open-source tools are integrated by the same technical interface through Amazon EMR:

* **Apache Kerberos** to configure Spark encryption and authentication. Benefit: increased security (reference: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-kerberos.html> )
* **Apache Hudi** to manage more efficiently change data capture (CDC) and helps with privacy regulations like GDPR and CCPA by simplifying record deletion. (reference: <https://aws.amazon.com/emr/features/hudi/> )
* **Apache Atlas** for the metadata classification and management (reference: <https://aws.amazon.com/blogs/big-data/metadata-classification-lineage-and-discovery-using-apache-atlas-on-amazon-emr/>)
* **Apache Ranger** for authorization, and configure fine-grained data access policies for databases, tables, columns, and S3 objects (reference: <https://aws.amazon.com/blogs/big-data/introducing-amazon-emr-integration-with-apache-ranger/>)

In the proposed architecture the architecture scales with respect to processing by adding new nodes to the Spark clusters. EMR supports automatic scaling, where the cluster can automatically add or remove nodes based on the workload demand (reference: <https://digital-alpha.com/running-heavy-workloads-with-aws-emr-and-spark/> ) .

In addition to the proposed tools to manage data processing, other tools have been considered:

* Apache Storm, open-source tool. Shortcoming: harder to operate and deploy compared to Spark (reference: <https://phoenixnap.com/kb/apache-storm-vs-spark> )
* AWS Lambda, third party tool. Shortcoming: even if the rating of the product is comparable with Spark, there is a vendor lock-in factor to be considered. Reference: <https://www.gartner.com/reviews/market/application-platforms-reviews/compare/product/aws-lambda-vs-spark-data-analytics-platform> . The tool might be considered in the future, in case the costs of managing the open-source tool will grow too much compared to the managed service or in case the usability of the open-source tool results being a constraint to the user adoption.

## Serving Layer

The serving layer provides a set of tools for end user data consumption: software components in this layer deliver their outputs to downstream data consumers or applications, where business value from the data is extracted and decisions based on the data are taken. Examples of those applications are the visualization of specific data by the medical facilities, the visualization of the personal data, such as their medical history by the patients or the prediction of bed availability or frauds avoidance for the medical facilities using ML models (reference: <https://experience.care/blog/big-data-and-ehr-data-analytics/> ).

In the serving layer there is no data stored since every data should be in the semantic layer of the storage, there the data is also encrypted, secured and access control management is implemented.

The data in the serving layer will be delivered through connectors to the visualization tools, such as Quicksight and Tableau. Moreover, the data it will be provided over connectors to the Data Science and AI tools such as Amazon SageMaker, IBM Warson Studio and Azure ML Studio. Finally, it will be queried by querying engines such as Amazon Athena.

# 8. Conclusion

The migration from the current infrastructure to the data lake architecture is a crucial step for the success of the Medical Data Processing Company, because it will cause major changes in all the business processes. It will allow not only to grant business continuity by removing the current bottleneck in the infrastructure but will also allow leveraging new business cases. Some advice for the success of this initiative:

* every time you get stuck along this journey think about the why you are doing it
* adopt agile methodologies to implement the changes
* empower the people driving these changes.

# 9. References

All links and references have been documented in the text.