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 |
| --- | --- | --- | --- |
| 01/07/2023 | Max Luong | 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 |
| --- | --- | --- | --- |
| FirstName LastName | Data Architect | Medical Data Processing | lmquanduc.iser@gmail.com |

# Note from Instructor:

# Consider this a comprehensive design document that you will deliver to the technical audience of the company.

# Provide detailed design and implementation level details

# You are expected to provide at least six pages worth of content (Does not include the cover (title) page and tracker page)

# Each section has guiding questions that will help you derive the responses.

# Purpose

**1.1 Purpose**

This document briefly presents a medical company's new big data architecture.

The document contains information from overall to details of the data architecture and technical requirements.

**1.2 Scope of this document:**

* In scope:
  + A design of high-level architecture, which is analyzed from the business use-cases
  + Documents to explain details about the architecture
  + A video to present the architecture to audiences
* Out scope:
  + Implementation details include codebase, configuration for data ingestion, data processing, and governance.

# Requirements

**2.1 Summary**

* Data Lake Solution Architecture Diagram, which adapts current business size and problems
* The data ingestion layer must be able to handle batch and streaming data.
* The data storage layer must have high availability, scalability, and flexibility characteristics.
* Data in the whole system ensures data quality and is ready to integrate with BI tools and external applications.

**2.2 Existing technical environment**

* 1 Master SQL DB Server
* 1 Stage SQL DB Server.
  + 64 core vCPU
  + 512 GB Ram
  + 12 TB disk space (70% full, ~8.4 TB)
  + 70+ ETL jobs running to manage over 100 tables.
* Three other smaller servers for Data Ingestion (FTP Server, data & API extract agents)
* Series of web and application servers (32 GB RAM each, 16 core vCPU)

**2.3 Current data volume**

* Data coming from over 8K facilities
* 99% of zip files size range from 20 KB to 1.5 MB
* Edge cases - some large zip files are as large as 40 MB
* Each zip file, when unzipped, will provide either CSV, TXT, or XML records
* In the case of XML zip files, each zip file can contain anywhere from 20-300 individual XML files, each XML file with one form.
* Average zip files per day: 77,000
* Average data files per day: 15,000,000
* Intermediate zip files per hour: 3500
* Intermediate data files per hour: 700,000
* Data Volume Growth rate: 15 – 20% YoY

**2.4 Business Requirements:**

* Improve uptime of the overall system
* Reduce latency of SQL queries and reports
* The design should be reliable and fault tolerant
* Architecture should scale as data volume and velocity increases
* Improve business agility and speed of innovation through automation and the ability to experiment with new frameworks
* Embrace open-source tools, and avoid proprietary solutions, which can lead to vendor lock-in
* Metadata-driven design - standard scripts should be used to process different types of incoming data sets rather than building custom scripts to process every data source.

Centrally store all of the enterprise data and enable easy access

**2.5 Technical Requirements:**

* Ability to process incoming files on the fly (instead of nightly batch loads today)
* Separate the metadata, data, and compute/processing layers
* Ability to keep unlimited historical data
* Ability to scale up processing speed with an increase in data volume
* The system should sustain a small number of individual node failures without any downtime
* Ability to perform change-data-capture (CDC), and UPSERT support on a certain number of tables
* Ability to drive multiple use cases from the same dataset without the need to move the data or extract the data
  + Ability to integrate with different ML frameworks such as TensorFlow
  + Ability to create dashboards using tools such as PowerBI, Tableau, or Microstrategy
  + Generate daily, weekly, and nightly reports using scripts or SQL
* Ad-hoc data analytics, interactive querying capability using SQL

# Data Lake Architecture design principles <approx. ½ page>

**3.1 Simple First**:

* The more complex your data lake is, the more difficult it will be.
* Keep it simple so everyone on your team can easily understand it.

**3.2 Single source of truth - data storage:**

* Ideally, you should store all your data in one place to easily access and analyze it. When you keep all your data in one place, it’s much easier to access and analyze.
* This can help you make better decisions about your business.
* Storing all your data in one place makes tracking and complying with regulations easier.

**3.3 Scalable:**

* Data lake needs to scale up or down as required, depending on how much data you store and process.
* A data lake must be scalable to handle the increasing volume of data businesses quickly generate today.
* If the data lake isn’t scalable, it will become overwhelmed and unusable. Ensure your data lake is designed for scalability from the beginning to avoid this problem.

**3.4 Secured:**

* One important consideration when designing your data lake is security. You must ensure that your data is safe and secure and that only authorized users can access it. Several ways to secure your data lake include encryption, role-based access control, and masking. Data encryption ensures that data is protected from unauthorized access. Role-based access control allows you to control which users can access which data. Data masking helps protect sensitive data by obscuring or hiding it from view.

**3.5. Right storage technology**:

* The type of storage technology you choose will affect the performance of your data lake. Choose the right one for your needs. Several cloud storage technologies can be used for data lakes. The most popular options are Amazon Simple Storage Service (S3), Azure Blob Storage, and Google Cloud Storage.S3 is a popular option because it offers much flexibility and scalability. You can use S3 to store any type of data, and it’s easy to scalable, so you can add more storage as needed. Azure Blob Storage is also a good option, especially if you need to store large files. It offers good performance and scalability, and it’s affordable. Google Cloud Storage is another good option, mainly if you already use other Google products. It provides good performance and reliability and is easy to use.

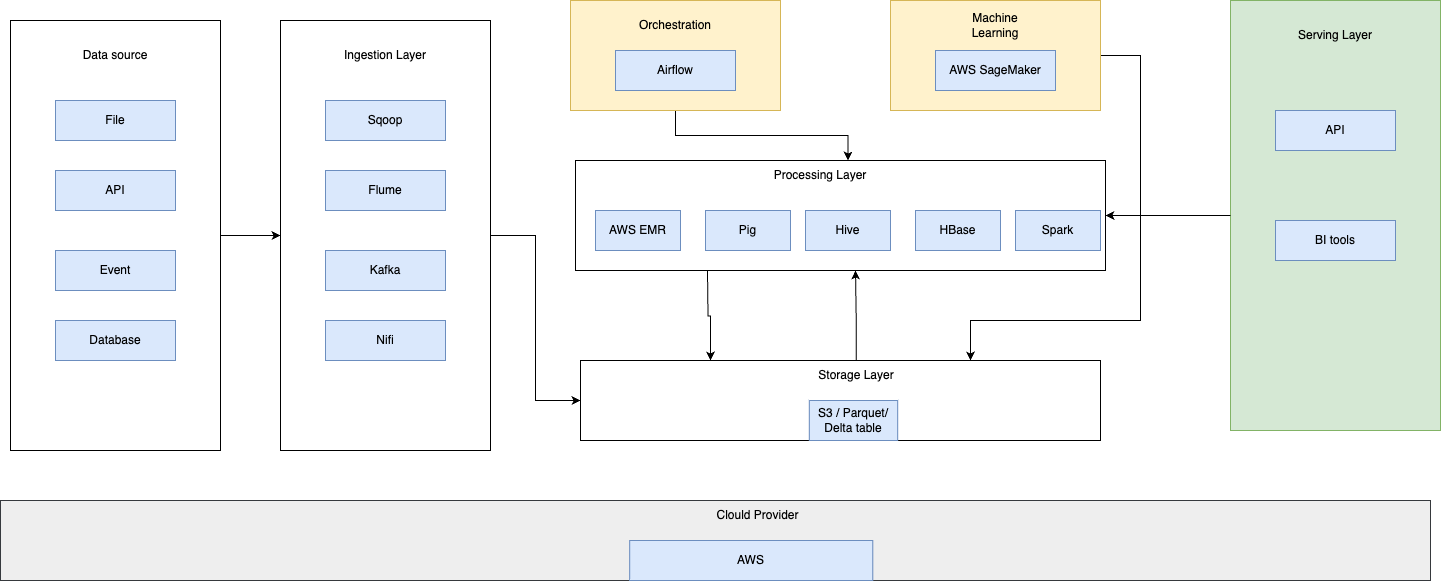
**3.6 Governance model early on:**

* Who will manage and govern the data in your data lake? Define this early on so there are no surprises later on. Several governance models can be used for data lakes. The most popular option is the centralized governance model. With this model, all the data in the data lake is centrally managed by a single team.
* This team ensures that the data is properly organized and labeled and that only authorized users can access it. Another popular governance model is the distributed governance model.
* With this model, the responsibility for managing and governing the data is distributed among several teams.
* Each team manages a specific subset of the data in the data lake. This can help ensure that different parts of the data lake are governed by other groups, improving overall efficiency.
* A third option is a self-governance model. With this model, each user is responsible for managing their data. This can be helpful if you want more control over your data, but it can also lead to chaos if everyone contains their data differently.
* Choose the suitable governance model for your needs and define it early in the design process.

# Assumptions

* Time to switch to data lake as soon as possible
* 100% of data will be moved to the cloud
* Building a data lake premise is possible here, but I want to develop architecture in the cloud. The goal of creating a data lake system architecture in the cloud can help scale quickly and take little time to maintain; many services will be automated, and no warranty fees.
* Cloud is preferred over on-premise infrastructure.
* An on-premise data lake imposes challenges. Companies must build their own data pipelines, pay the ongoing management and operational costs in addition to the initial investment on servers and storage equipment, and manually add and configure their servers to scale a data lake to cater to moving users or increasing data volume.

# Data Lake Architecture for Medical Data Processing Company



# Design Considerations and Rationale

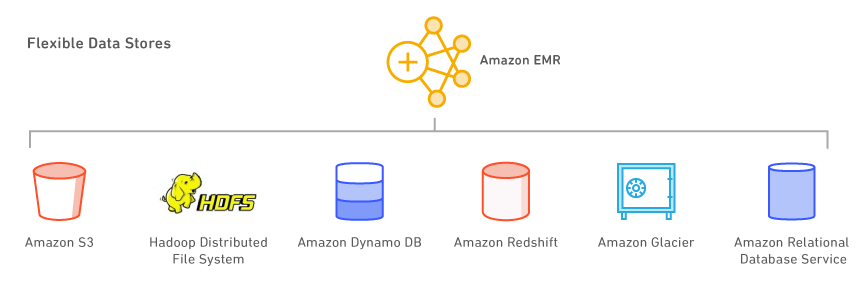
## **Ingestion Layer**

* Data ingestion is importing large, assorted data files from multiple sources into a single, cloud-based storage medium—a data warehouse, data mart, or database—where it can be accessed and analyzed. Data may be in multiple forms and come from hundreds of sources; it is sanitized and transformed into a uniform format using an extract/transform/load (ETL) process.
* I considered that: Apache Sqoop, Flume, Kafka, Debezium, and Nifi, but I decided to use Amazon services:
  + Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL and Oracle to Hadoop HDFS and export it to relational databases from the Hadoop file system. This is a brief tutorial that explains how to make use of Sqoop in the Hadoop ecosystem.
  + Flume is a tool/service/data ingestion mechanism for collecting, aggregating, and transporting large amounts of streaming data, such as log files, events (etc. . .) from various sources to a centralized data store. Flume is a highly reliable, distributed, configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.
  + Kafka is designed for distributed high throughput systems, can handle a high volume of data, and enables you to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss.
  + Debezium is an open-source distributed platform for change data capture. Debezium lets your apps react every time data changes, and you don’t have to change your app that modifies the data. Debezium continuously monitors your databases and lets any applications stream every row-level change in the same order they were committed to the database.
  + Nifi is a real-time data ingestion platform that can transfer and manage data transfer between sources and destination systems. It supports various data formats like logs, geo-location data, social feeds, etc. It also supports many protocols like SFTP, HDFS, Kafka . . . This support for various data sources and protocols makes this platform popular in many IT organizations.

## Storage Layer

* Amazon S3 provides the foundation for the storage layer in our architecture. Amazon S3 provides virtually unlimited scalability at a low cost for our serverless data lake. Data is stored as S3 objects organized into raw, cleaned, and curated zone buckets and prefixes. Amazon S3 encrypts data using keys managed in AWS KMS. IAM policies control granular zone-level and dataset-level access to various users and roles.
* Amazon S3 provides 99.99% of availability and 99.999999999% of durability and charges only for the data it stores. Amazon S3 offers colder-tier storage options called Amazon S3 Glacier and S3 Glacier Deep Archive to reduce costs significantly. To automate cost optimizations, Amazon S3 provides configurable lifecycle policies and S3 Intelligent-Tiering options to automate moving older data to colder tiers. AWS services in our ingestion, cataloging, processing, and consumption layers can natively read and write S3 objects. Additionally, hundreds of third-party vendors and open-source products and services provide the ability to read and write S3 objects.
* Amazon S3 is a petabyte-scale object store providing virtually unlimited data storage scalability. You can store structured data (such as relational data), semi-structured data (such as JSON, XML, and CSV files), and unstructured data (such as images or media files).
* Any structure (including unstructured data) and format can be stored as S3 objects without predefining schema. This enables services in the ingestion layer to quickly land various source data into the data lake in its source format. After the data is ingested into the data lake, components in the processing layer can define the schema on top of Amazon S3 datasets and register them in the cataloging layer. Services in the processing and consumption layers can then use schema-on-read to apply the required structure to data read from S3 objects. Datasets stored in Amazon S3 are often partitioned to enable efficient filtering by services in the processing and consumption layers.
* Hadoop Distributed File System (HDFS): [HDFS](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html) is the Hadoop file system. Amazon EMR’s current topology groups its instances into three logical instance groups: Master Group, which runs the YARN Resource Manager and the HDFS Name Node Service; Core Group, which runs the HDFS DataNode Daemon and the YARN Node Manager service; and Task Group, which runs the YARN Node Manager service. Amazon EMR installs HDFS on the storage associated with the instances in the Core Group.

## Processing Layer



* Amazon EMR (previously known as Amazon Elastic MapReduce) is an Amazon Web Services (AWS) tool for big data processing and analysis. Amazon markets EMR as an expandable, low-configuration service that provides an alternative to running on-premises cluster computing.
* Amazon EMR is based on Apache [Hadoop](https://www.techtarget.com/searchdatamanagement/definition/Hadoop), a Java-based programming framework that supports processing large data sets in a distributed computing environment. Using MapReduce, a core component of the Hadoop software framework, developers can write programs that process massive amounts of unstructured data across a distributed cluster of processors or standalone computers. Google developed it for indexing webpages and replaced its original indexing algorithms and [heuristics](https://www.techtarget.com/whatis/definition/heuristic) in 2004.
* Amazon EMR processes big data across a Hadoop cluster of virtual servers on Amazon Elastic Compute Cloud ([EC2](https://www.techtarget.com/searchaws/definition/Amazon-Elastic-Compute-Cloud-Amazon-EC2)) and Amazon Simple Storage Service (S3). The *Elastic* in EMR's name refers to its dynamic resizing ability, which enables administrators to increase or reduce resources depending on their current needs.
* Amazon EMR is used for data analysis in log analysis, web indexing, data warehousing, [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML) (ML), financial analysis, scientific simulation and bioinformatics. It also supports workloads based on Apache Spark, Apache Hive, Presto, and Apache HBase, which integrates with Hive and Pig, open-source data warehouse tools for Hadoop. Hive uses queries and analyzes data, and Pig offers a high-level mechanism for programming MapReduce jobs to be executed in Hadoop.
* [Apache Spark](https://aws.amazon.com/big-data/what-is-spark/) is an engine in the Hadoop ecosystem for the fast processing of large data sets. It uses in-memory, fault-tolerant resilient distributed datasets (RDDs) and directed acyclic graphs (DAGs) to define data transformations. Spark also includes Spark SQL, Spark Streaming, MLlib, and GraphX. Learn [what](https://aws.amazon.com/big-data/what-is-spark/) Spark is and more about [Spark on EMR](https://aws.amazon.com/emr/details/spark/).
* [Apache Flink](https://flink.apache.org/) is a streaming dataflow engine that makes it easy to run real-time stream processing on high-throughput data sources. It supports event time semantics for out-of-order events, exactly-once semantics, backpressure control, and APIs optimized for writing streaming and batch applications. Learn more about [Flink on EMR](http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/emr-flink.html).
* [TensorFlow](https://www.tensorflow.org/) is an open-source symbolic math library for machine intelligence and deep learning applications. TensorFlow bundles multiple machine learning and deep learning models and algorithms and can train and run deep neural networks for many use cases. Learn more about [TensorFlow on EMR](https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-tensorflow.html).
* Amazon SageMaker is a cloud-based machine-learning platform that helps users create, design, train, tune, and deploy [machine-learning](https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-machine-learning) models in a production-ready hosted environment. The AWS SageMaker comes with a pool of advantages.

## Serving Layer

This layer queues batch views prepared by the batch layer and then indexes them. The serving layer’s goal is to make the data queryable in a very short period. The server layer stores the output and merges the batch layer output with the speed layer output.

BI tools:

* Apache Superset is fast, lightweight, intuitive, and loaded with options that make it easy for users of all skill sets to explore and visualize their data, from simple line charts to highly detailed geospatial charts.
* Microsoft Power BI is a [business intelligence](https://www.techtarget.com/searchbusinessanalytics/definition/business-intelligence-BI) (BI) platform that provides nontechnical business users with tools for aggregating, analyzing, visualizing, and sharing data. Power BI's user interface is fairly intuitive for users familiar with [Excel](https://www.techtarget.com/searchenterprisedesktop/definition/Excel), and its deep integration with other Microsoft products makes it a versatile self-service tool that requires little upfront training.

Data product:

* A data product is a reusable data asset built to deliver a trusted dataset for a specific purpose. It collects data from relevant data sources — including raw data — processes it, ensures data quality, and makes it accessible and understandable to anyone who needs it to meet specific needs. Data scientists and analysts analyze data products to inform predictive analytics, build data models, build new reports, assist in machine learning, and more.

# 8. Conclusion

* For storage, using S3 storage is cost-effective, durable, and scalable storage.
* Many solutions ingest data into a data lake, from streaming (CDC, Kafka, Spark Streaming, Flink, ..) to batching FSTP, buck loading;...
* For processing and analyzing data, EMR, Spark, SageMaker, and any analytical solution can rapidly be scaled to power any extensive data application.
* In the serving layer,, we expose insight by BI tools or custom applications

# 9. References

* Data lake design principles: <https://vitalflux.com/data-lake-design-principles-best-practices>