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 |
| --- | --- | --- | --- |
| 10/14/2022 | Thu Huynh | 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 |
| --- | --- | --- | --- |
| Thu Huynh | Data Architect | Medical Data Processing | student@email.com |

# Purpose

**Purpose**

We were asked to create this document by MDPC's CTO since the company has encountered various problems with processing data with its current unstable and easy-to-crash data architecture. As a result, the CTO is seeking for an alternate design.

The goal of this document is to give a thorough data lake design architectural proposal to the Medical Data Processing Company (MDPC).

This project aims to rebuild and enhance the data layer, which includes data ingesting, processing, storing, and serving.

**Content Outline:**

* Purpose and scopes of work
* Business and Technical requirements
* Data Lake Architecture design principles
* Design Considerations and Rationale
* Conclusion
* References

**Target Audience**

This document is intended for the company's more technically focused architects, software engineers, and tech leads.

In Scope and Out of Scope Items

• In scope: the requirements, the assumptions, the design of the data architecture

• Out of scope: implementation of the data architecture, data governance, machine learning

The items that are in scope for this document are the various components of my architectural design, arguments defending those decisions, as well as explanations of the requirements that are driving those decisions. Some elements that are out of scope for this document are the frameworks we use for the analytics/dashboard layers, the implementation of the data warehouses for the analytics layer, and the application code for the applications supporting our clients.

**Out of Scope:**

This document has the proposal and the architecture details and requirements, but it doesn’t have considerations about the implementation part of it, which would happen to be in a separate document and effort.

Handling JSONS is not a requirement on this project, the proposed solution would be able to handle it though.

It is out of scope for this project to have a transactional DB but it could be added

It is out of scope for this project to have a data warehouse but it could be added.

**In scope**

* All details about new Data Lake architecture proposed solution with all layers included
* Suggested tools/frameworks and technologies that help to implement the proposed architecture
* Visual of Data lake diagram

**Out of Scope**

* The detailed plan about actual implementation of this proposed architecture, such as financial plan, human resources needed or timeline.
* The implementation/improvements for analytics/dashboard layers
* The code/solution for other supporting applications that Medical Data Processing Company is working with

# Requirements

**Problems:**

Company currently hosts over 8TB of data in SQL Server today. Company has experienced hyper growth over the past 3 years. However, as the volume of data continues to grow, the existing single node SQL Server is not able to scale.

Last week, there was a surge in data, during the nightly ETL process, the database server crashed and the whole system was offline for several hours .Since the SQL database does not have enough capacity, doing analytics and reporting requires the company to export the required data on a nightly basis to seperate servers.

### 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
* 3 other smaller servers for Data Ingestion (FTP Server, data and API extract agents)
* Series of web and application servers (32 GB RAM Each, 16 core vCPU)

### Current Data Volume

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

### Business Requirements

* Improve uptime of overall system
* Reduce latency of SQL queries and reports
* System 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 ability to experiment with new frameworks
* Embrace open source tools, avoid proprietary solutions which can lead to vendor lock-in
* Metadata driven design - a set of common scripts should be used to process different types of incoming data sets rather than building custom scripts to process each type of data source.

Centrally store all of the enterprise data and enable easy access

### 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 increase in data volume
* System should sustain small number of individual node failures without any downtime
* Ability to perform change data capture (CDC), UPSERT support on a certain number of tables
* Ability to drive multiple use cases from 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, nightly reports using scripts or SQL
* Ad-hoc data analytics, interactive querying capability using SQL

All mentioned technical environments, business and technical requirements are described in company profile - problem statements provided in this project

# Data Lake Architecture design principles

**High Availability**

We need to store data in every forms: from raw to processed, clean data with many version of data in each step of ETL in separate locations. This way we can always dig into what’s happening in each step and retry any actions that fail

Then data also needs to be stored in more than one zone/server in case one of them crashes, there will always be backup version to keep data alive for any operations

**Unlimited Scalability**

With the fast growth of this company, the new data lake architecture must be able to scale up or down according to the needs of the business. As of now, vertical scalability cannot handle the increased data, horizontal scaling will give us the ability to handle bigger amounts of data with more stability.

**Optimal Performance**

Except for real-time data read and write, Medical Data Company also has a crucial requirement for real-time analytics. This requires the new design must have high performance

**Data Governance**

Data collected is used for various stakeholders of Medical Data Company, from internal staff to top leaders and also their customers. This is why we need to make sure to establish and implement clear and strict access policies for each stakeholder - for the sake of data security.

# Assumptions

**Cloud storage**

All data will be stored in the cloud, not on-premise. Amazon will be the provider for most of the tools, services that are proposed in this document, including S3 and Redshift.

**Cloud Computing**

All computing actions will be implemented using cloud computing services, for less risks and more secure, more available with cloud computing services - easy to scale and always available

**Historical data**

Historical data is assumed to be migrated by a professional team to cloud

**Integration with existing reporting layers**

It is assumed that this new data lake solution can be integrated with the reporting layers that the current company is using and be handled by a professionals who can advised and give directions for guidelines to integrate.

**Programming Language**

The solution proposed will include using Spark - a data processing framework that supports multiple programming language. The solution proposed in this document will use the language that Spark supports.

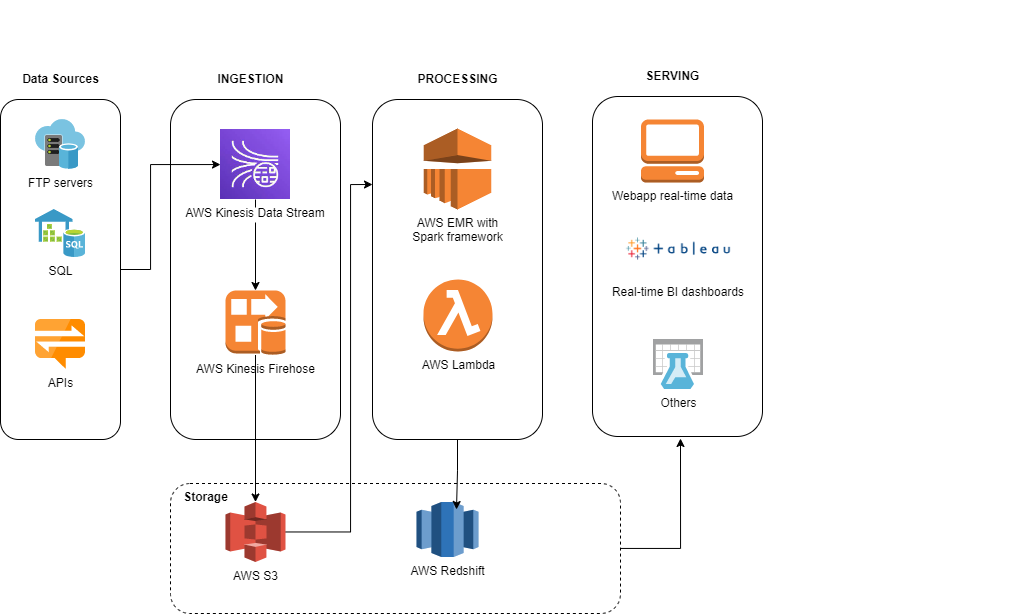
**Employee Skills**

We will assumpt that the company has a team to be trained and implement the proposed solution and later can use and monitor the new infrastructure effectively.

\*\* **Potential risks:**

* *For programming language: There can be a* ***potential risk*** *is that Spark does not support the programming language that the current company staff is using.*
* *For cloud storage service: there is a* ***potential risk*** *that S3 or Redshift does not support the connection to some existing applications that the current company is using. In this case, the technical team will discuss other alternatives on how to acquire data from these apps to our cloud storage.*
* *If the existing technical team of the company is not capable of being trained and contributing to the implementation of this solution, there is a risk that the project cost will be bigger and more expensive, in terms of time, people and finance.*

# Data Lake Architecture for Medical Data Processing Company



# Design Considerations and Rationale

## Ingestion Layer

As we go with the solution to use data lake approach, all data coming from Databases, FTP Servers, APIs will be migrated to our data lake. The ingestion layer is designed to handle both SQL and NoSQL, or any other data/ file format. With the S3 storage service from AWS, we can store all objects with high availability and easy to scale up or down.

The flow is any different data types will be captured and collected through **AWS Kinesis Data Stream** - the real-time data streaming service in Amazon Kinesis with high scalability and durability. It can help in continuously capturing multiple gigabytes of data every second from multiple sources, then writing to AWS Data Firehose using Data Stream.

**AWS Kinesis Data Firehose** provides the facility of loading data streams into AWS data stores. Kinesis Data Firehose provides the simplest approach for capturing, transforming, and loading data streams into AWS data stores - which is S3. Streaming data is delivered to a S3 bucket. We can optionally back up source data to another Amazon S3 bucket.

Third-party tools/ data sources may be occasionally requested to load into our data lake. In these cases, the loading method will be evaluated accordingly and data will be ingested and stored in S3.

There are other open-source tools which can perform the same function such as Apache Kafka to handle streaming data or Apachi Nifi to handle data flow and migration. However, we strongly prefer using the managed AWS services for the higher availability, automatical scale in minutes and no more managing on-premise servers or hardwares.

## Storage Layer

There are so many advantages of cloud data lake. First of all, it’s about high scalability - which used to be a critical issue with Medical company since data amount is growing day by day and when the existing architecture can’t no longer handle more data, all problems come.

Scalability is the measure to increase or decrease the resource as per need. We can add and grow as much as we want, with no restrictions. We can have different buckets for different functions, with different security setups.

AWS S3 has availability zones or regions across multiple countries to ensure high availability as well. It gives every user, its service access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of websites. With Amazon S3, we only pay for the data we use.

By this way, in the future, when the company grows 20% or more, the system is always able to meet the company needs.

Although modern data lake stores have high availability, we can still use cross-regional storage for disaster recovery. As for back-up, we will use the high availability deployments as well as continous backup across multiple regions.

We will also store all details about metadata information in our data lake, but in 1 specific schema in the bucket. Metadata should include all details about schemas, data types, data sources, data size and frequency of data loading.

AWS also offers a great solution for data security in AWS Identity and Access Management Roles. We categorize access for users in different groups with defined roles and allowed actions, then set up individual user accounts with AWS IAM. That way each user group is given only the permissions necessary to fulfill their job duties. Other actions to secure our data can be: use multi-factor authentication with each account, use SSL/TLS to communicate with AWS resources, use AWS encryption solutions, or use advance security services such as AWS Macie

The plan is having 3 different S3 buckets for different purposes:

* 1 bucket for storing raw data that comes from different data sources
* 1 bucket for staging layer - which only stores data that is needed to do some transformation
* 1 bucket for transformed data

Furthermore, SQL data is crucial for other layers and also for company analysis work, AWS Redshift is added to the storage. AWS Redshift is a fully managed, scalable cloud data warehouse that accelerates our data to insights with fast, easy, and secure analytics at scale, analyze data from terabytes to petabytes and run complex analytical queries.

## Processing Layer

For processing layer, from small data amount to big data, Python and Spark is strongly recommended for simple to powerful ETL jobs.

Apache Spark is a distributed processing framework and programming model that helps do machine learning, stream processing, or graph analytics using Amazon EMR clusters. Similar to Apache Hadoop, Spark is an open-source, distributed processing system commonly used for big data workloads. However, Spark has several notable differences from Hadoop MapReduce. Spark has an optimized directed acyclic graph (DAG) execution engine and actively caches data in-memory, which can boost performance, especially for certain algorithms and interactive queries.

Spark natively supports applications written in Scala, Python, and Java. It also includes several tightly integrated libraries for SQL (Spark SQL), machine learning (MLlib), stream processing (Spark streaming), and graph processing (GraphX). These tools make it easier to leverage the Spark framework for a wide variety of use cases.

We can install Spark on an Amazon EMR cluster along with other Hadoop applications, and it can also leverage the EMR file system (EMRFS) to directly access data in Amazon S3. Hive is also integrated with Spark so that you can use a HiveContext object to run Hive scripts using Spark.

For real-time data, Spark on AWS EMR consumes and processes real-time data from Amazon Kinesis, Apache Kafka, or other data streams with Spark Streaming on EMR. Perform streaming analytics in a fault-tolerant way and write results to S3 or on-cluster HDFS.

EMR clusters also can run Spark in batch mode to help process batch data or CDC. AWS Lambda is a compute service that lets us run code without provisioning or managing servers. Lambda runs our code only when needed and scales automatically, from a few requests per day to thousands per second. After our Lambda function code for necessary data transformation, we can launch the function to initiate the creation of a transient EMR cluster. It will run the Spark job and terminate automatically when the job is complete.

There aren't many 3rd party tools that have been taken into consideration. This is due to the fact that our proposed architecture is adaptable enough to support the deployment of a wide variety of tools for processing. The majority of cloud service providers will, at the very least, let us deploy some other processing frameworks such as Hive, Spark, Hadoop MapReduce, Pig, etc.

Data after processed will be loaded back to S3 in the proper bucket and Redshift. With Amazon EMR release versions 6.4.0 and later, every Amazon EMR cluster created with Apache Spark includes a connector between Spark and Amazon Redshift.

## Serving Layer

All the outputs from processing layers will be the inputs for many objectives: serve the transformed data to business users/ end customers at the serving layer or serve the data for other applications. Here, our serving layer is considered as analytical serving layer, where we use a data warehouse with a dimensional model to enable complex queries and connections to multiple applications.

Data consumers, who can divided into different stakeholders:

* End customers of medical company: who have the need to view/ track their registered information with us, through our public website/app.
* Directors/managers/company leaders: who need the transformed and correct data for real-time dashboards/reports, for analyzing and make decisions for business matters
* Internal team and departments: such as Business department need data for their revenue reports, or Marketing team for their monitoring and evaluation on implemented marketing activities. Data team for building dashboards to answer business questions and so on.
* Third-party tools: where our data is needed for them to perform other required tasks accordingly

# 8. Conclusion

Thorough this document, we proposed a high available, secured, scalable, low-cost data lake architecture that accommodates to the growing data needs of Medical company. This architecture will include 4 mentioned layers: ingestion, storage, processing and serving, where all the cloud services are chosen to implement.

The ingestion layer of AWS Kinesis will collect large amount of real-time data as well as batch data and load to the storage layer.

The storage layer of S3 and Redshift can meet all the needs of storing all types of data format and size, easy to scale up or down automatically and a data warehouse that can handle from simple to complex queries with multiple connectors to other applications/frameworks.

The processing layer with AWS EMR and Spark framework with in-memory processing will ensure the fast processing time to handle big data, then load back to S3 and a data warehouse of Redshift.

The serving layer will hereby connect to the processing layer with supporting connectors to load transformed data to BI apps, websites or other third-party applications.

# 9. References

<https://aws.amazon.com/s3/faqs/>

<https://www.cloudzero.com/blog/aws-redshift>

<https://aws.amazon.com/emr/>

<https://docs.aws.amazon.com/firehose/latest/dev/writing-with-kinesis-streams.html>

<https://www.whizlabs.com/blog/aws-kinesis-data-streams-vs-aws-kinesis-data-firehose/>

<https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/launch-a-spark-job-in-a-transient-emr-cluster-using-a-lambda-function.html>

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark.html>