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 |
| 06/09/2024 | XXXXXX | 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 |
| XXXXXXX | Data Architect | Medical Data Processing |  |

# Purpose

There are many problems/issues with current system : the existing single node SQL Server is not able to scale , there was a surge in data last week, there is no rapid back-up and recovery plan,… This document is created to propose renewal Data Architect in Medical Data Processing Company.

This document contains:

* Requirements
* Data Lake Architecture design principles
* Assumptions
* Data Lake Architecture for Medical Data Processing Company
* Design Considerations and Rationale
* Conclusion

I created this document to present new Data Architect to solve current system issues, and to show capability of new system with unlimited scale, high availability, cost controlled, easily to adapt with new data format.

This document is for a technical audience who is interested in your design ideas and decisions at a deep level.

Scope Items:

* New Data Architect design to innovate current system.
* Migrate old data from current system to new system.
* New Data Lake as central data storage.
* Data pipeline with scheduling to load, transform, extract data.

Out of scope items:

* Update front end source code. For example: add responsive feature to screen, design new UI/UX for end users.
* Maintain high availability of external system connection.
* Unit test script

# Requirements

**Current problems:**

Currently company has experienced hyper growth over the past 3 years. However, some parts of current system were not designed to scale. And ETL processes and SQL reporting queries are running slow due to increased data volumes.

Also, there is no rapid back-up and recovery plan. There are many data silos in the company, making hard to track the latest version of data from many sources.

**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

These requirements come from many best practices of other enterprise system. Theses are challenges need to face when we design new Data architect system.

# Data Lake Architecture design principles

Building a data lake can break down data silos and create a central data repository empowers organizations to make data-driven decisions and innovate quickly.

Some Data Lake Architecture design principles:

* Size and location of data: because company has experienced hyper growth over the past 3 years, size of data will increase fast. We need a central place to store unlimited data with high availability.
* Security and compliance: theses are always most important things to consider when designing new system. We need high security to prevent precious data from malicious people. And to comply with rules/compliances, we can provide legal protection, strengthens data security, and enhances an organization's reputation.
* Access query patterns and needs: we need to consider what query patterns are used, to design data architect to match the need of users.
* Build a decoupled design with separate compute, storage, and processing layers: decoupled help to scale storage and compute independently, to process data fast and efficiently.
* DevOps automation. automated as much as possible: automation help bringing new data, pipeline into system fast and less error prone.
* File format: current system is using XML, TXT, CSV as file format, these formats are row-based format and are not suitable for Big Data system. We need to convert into new format with column-based to process data effectively and efficiently.

# Assumptions

* Company employees are ready to change current system, and to learn new tools to operate with data.
* Data source from external system is always available.
* Current SQL Server have enough capacity and CPU processing to migrate into new data system.
* The data encoding in SQL Server can be convert into UTF-8 safely.

Potential risks with these assumptions:

Because data from external system can be down anytime, we should create plan to handle this situation. We can postpone getting data from these sources and process it when data becomes available. Or we can discard this data, and run pipeline as backfill to process old data later.

Current SQL Server maybe do not have enough CPU to support migrate process. We can increase RAM of SQL Server machine, or we use the latest backup file to migrate first, then use SQL server to support migrate the new data after backup snapshot.

# Data Lake Architecture for Medical Data Processing Company

A diagram of a software company

Description automatically generated

# Design Considerations and Rationale

## Ingestion Layer

**How do you plan to ingest different types of data?**

I use different tools for different types of data. For example: Apache Kafka, Apache Sqoop, AWS CLI. Each type of data is bound with each tool.

**How would you ingest data coming from Databases, FTP servers, APIs?**

* For files sent into FTP Servers, we trigger cronjob to copy file into S3 by using rsync command.
* For Data in Databases need to move into S3, we use Apache Sqoop to transfer data.
* For CDC Data in Databases, we send this CDC Data into Apache Kafka, then use Kafka connect S3 connector to move data into S3.
* For user activity log or data from IOT devices, we stream this data into Apache Kafka, then move into S3.
* For APIs call, we use API Gateway to receive request, then we use lambda to send this Log/Data into S3.

**What tools would be used? Why?**

* Apache Kafka: Apache Kafka is an open-source distributed streaming system used for stream processing, real-time data pipelines, and data integration at scale. Kafka have capable of handling extremely large number of messages per day.
* Apache Sqoop: Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. I can move large amount of data from Database into Data Lake as Raw data.
* API Gateway: can receive and process very large number of request concurrently.

**How would the ingestion layer design scale?**

Because the tools we are using were designed for Big Data system, they can easily scale and handle large amount of data. For example: Apache Kafka, Apache Sqoop.

For API gateway, it is serverless service and can serve a very large amount of request in a same time.

**What other tools were considered? (3rd party tools, open source tools considered but did not make it to the architecture you are proposing). Are there other shortcomings to your selection of tools? If so what? Does the 3rd party tool solve that?**

RabbitMQ: we can use RabbitMQ for ingest stream data, but Apache Kafka is more suitable to this architect. Apache Kafka is designed to handle log data with high through put and high durability.

## Storage Layer

**How do you plan to store a vast amount of data?**

We will use AWS S3 for store a vast amount of data. Amazon S3 is an object storage service offering unlimited scalability, data availability, security, and performance. We can use S3 as a centralized repository that allows us to store all structured and unstructured data at any scale.

**How would the system handle 20% YoY Data Growth rate?**

In ingestion layer and storage layer, we are using tools and services with high scalability, out system can handle 20% YoY Data Growth rate.

**How do you plan to handle back-up and recovery? What are the strategies?**

Data is stored in AWS S3 as standard storage can have 99.999999999% durability and 99.99% availability of objects over a given year.

And this data is replicated between many Availability zones in chosen region. We can config to copy data to another region as backup.

**How do you plan to store custom metadata information? What type of information would metadata hold?**

We can store custom metadata in DynamoDB. This metadata hold information that describes and explains data. It provides context with details such as the source, type, owner, and relationships to other data sets.

**What format of the data do you plan to use? Why?**

For raw data, I keep data format as-is, for example : XML, TXT, CSV. For data in the Lake House, I use Parquet as base file format for Apache Hudi.

**How do you plan to secure data (at a high-level)? Identify 2-3 techniques/tools/considerations**

We are using AWS S3 for Data Lake. AWS S3 provide many features for security.

We can use IAM role for grant access/block access into S3.

We can use S3 Object Lock feature to prevent objects are modified by others.

We can use encryption in both in transit and at rest.

**What other tools were considered? (3rd party tools, open source tools considered but did not make it to the architecture you are proposing). Are there other shortcomings to your selection of tools? If so what? Does the 3rd party tool solve that?**

We can use HDFS in place of AWS S3. But I choose AWS S3 because:

* S3 is fully managed service by AWS, make it easily for user to use.
* S3 is more scalable than HDFS.
* S3 is more cost-efficient and likely cheaper than HDFS.
* Data in S3 is always persistent, unlike data in HDFS.

## Processing Layer

**How do you plan to process the data?**

I will use Apache spark for data processing tool, and Apache Airflow to schedule and workflow orchestration tool.

**How do you satisfy different processing needs? Batch, Realtime, CDC?**

* For Batch processing: I use Apache Airflow to trigger Spark job to process data in batch. Ex: every hour, every day, …
* For Realtime processing: I use micro-batch processing with Spark to process Realtime data. Also, I can use Spark Streaming to process.
* For CDC: CDC data comes from databases into Kafka, then processed by Apache Spark.

**How do you enable ad-hoc querying capabilities?**

We can use AWS Athena to ad-hoc query into AWS S3. We can query raw data files in XML, TXT, CSV format, and we can also query Hudi datasets in Parquet format.  
**What different tools are involved for processing?**

Apache Spark: Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytics, machine learning,

Apache Airflow: is an open-source tool and framework for running your data pipelines in production. As an industry-leading data workflow management tool, Apache Airflow leverages Python to allow data practitioners to define their data pipelines as code.

**What other tools were considered? (3rd party tools, open source tools considered but did not make it to the architecture you are proposing). Are there other shortcomings to your selection of tools? If so what? Does the 3rd party tool solve that?**

We can use Apache Hive in place of Apache Spark, but I chose Apache Spark because:

* Apache Spark has rapidly emerged as the de facto standard for big data processing across all industries and use cases
* Because Spark use RAM to store intermediate data, Spark can be up to 100 times faster than Hadoop-based processing like Hive.

**How does the proposed architecture scale with respect to processing?**

With high load, Apache Spark can scale very fast to process large amount of data with Massive Data Processing Capacity.

Apache Airflow can also easily scale horizontally.

## Serving Layer

Serving layer has responsibility to expose the views created by both the batch and speed layer for querying by other systems or users. For using in Machine Learning, Data Analytics, Reporting.

About type of data will be stored, I will store curated, well processed data. For each business purpose, I will store data in each data mart tables for user systems, users to consume.

Data in the serving layer can be used for:

* For machine learning: processed data will be used as input for TensorFlow to predict future data.
* For data analytics, reporting: Tableau and PowerBI can query data and generate reports to provide insight of data.

# 8. Conclusion

This document is providing solution to solve current problems of “Medical Data Processing” Company. This new data architect system can scale easily with massive growing data, provide a central data storage as single of truth, … It can also support new data type in future.

# 9. References

Apache Hudi: <https://hudi.apache.org/>

Apache Spark: <https://spark.apache.org/>

Apache Airflow: <https://airflow.apache.org/>