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 |
| 13/05/2023 | Kien Dang | 0.1 | Initial draft |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  |  |  |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
| Kien Dang | Data Architect | Medical Data Processing | kiendt20@fpt.com |

# Note from Instructor:

# Consider this as 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 6 pages worth of content (Does not include the cover (title) page and tracker page)

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

# Purpose

The purpose of the document to provide detail technical design proposal for an enterprise data lake system.

The document contains a detailed technical design proposal for an enterprise data lake system. It’s includes descriptions of the architecture, explanations of how the proposed design can solve the company’s challenges. And document should clearly state any assumptions or potential risks to the design.

The document is intended to showcase data architecture ability to design a technical solution for an enterprise data lake system and to demonstrate understanding of the business problem and recommendation for a solution.

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

In scope: a design of data lake system architecture that can handle large volumes of data, integrating various data sources, providing a flexible and scalable solution that can meet computer’s needs, a detailed explanation of the proposed solution, including the technologies and tools used.

Out of scope:

# Requirements

<Summary of requirements for Data Lake. Summarize your understanding of the problem statement. >

Summary:

* Design a system with high availability, reliability, resiliency.
* Easily Scalable with increasing in data volume and velocity.
* Ad-hoc data analytics, interactive querying capability using SQL.
* Integrate flexibly with report, dashboards, ML frameworks.

<Existing Technical Environment>

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 & API extract agents)
* Series of web and application servers (32 GB RAM each, 16 core vCPU)

<Current Data Volume>

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: 3,500**
* **Average data files per hour: 700,000**
* **Data Volume Growth rate: 15 – 20% YoY**

<Business Requirements>

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
* 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>

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

<Where do you find these requirements? Have you seen them somewhere before? >

Where do you find these requirements: From company profile problem statement file.

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

<List of the design principles. What is the baseline criteria to design the system? What rules/guiding principles should be followed?>

**Scalability & Flexibility**: Medical Data System will be dealing with large amounts of data, and as the amount of data grows, the system needs to be able to scale up or down as needed. And with dealing with different types of data, including (CSV, TXT, XML . . .). By designing a flexible architecture, Medical Data Systems can ensure that their data lake can handle any type of data that they may encounter.  
But when change format file from CSV, XML, TXT to Avro and Parquet which are standard, well-known and can accessible by different tools may good for performance.

**Resilience**: Use a central meta-data repository such as AWS Glue. This will allow data architecture to centralize and manage all meta-data in a single location, reducing operational costs in infrastructure, IT resources and engineering hours. Event sourcing should be used and an immutable log of all incoming events and maintained on object storage. Event sourcing enables data engineer to retrace the steps to learn about the extract transformation applied on the raw data.

**Easy to use**: By automating the ETL pipelines that ingest the raw data and perform the relevant transformations per use case we can prevent the data engineering bottleneck that might form if relying on coding-based ETL frameworks such as Apache Spark.

**Performance:**

* Every file stored need contain meta-data to understand data structure.
* Use columnar file formats: Apache Parquet and ORC
* Keep data in optional file sizes, A hot/cold architecture is recommended like hot for frequency files, cold for un-frequency files
* Use efficient partitioning strategy.

# Assumptions

<What are the assumptions you have made while creating the Data Lake architecture?>

<Be creative, what questions did you have while designing the architecture?>

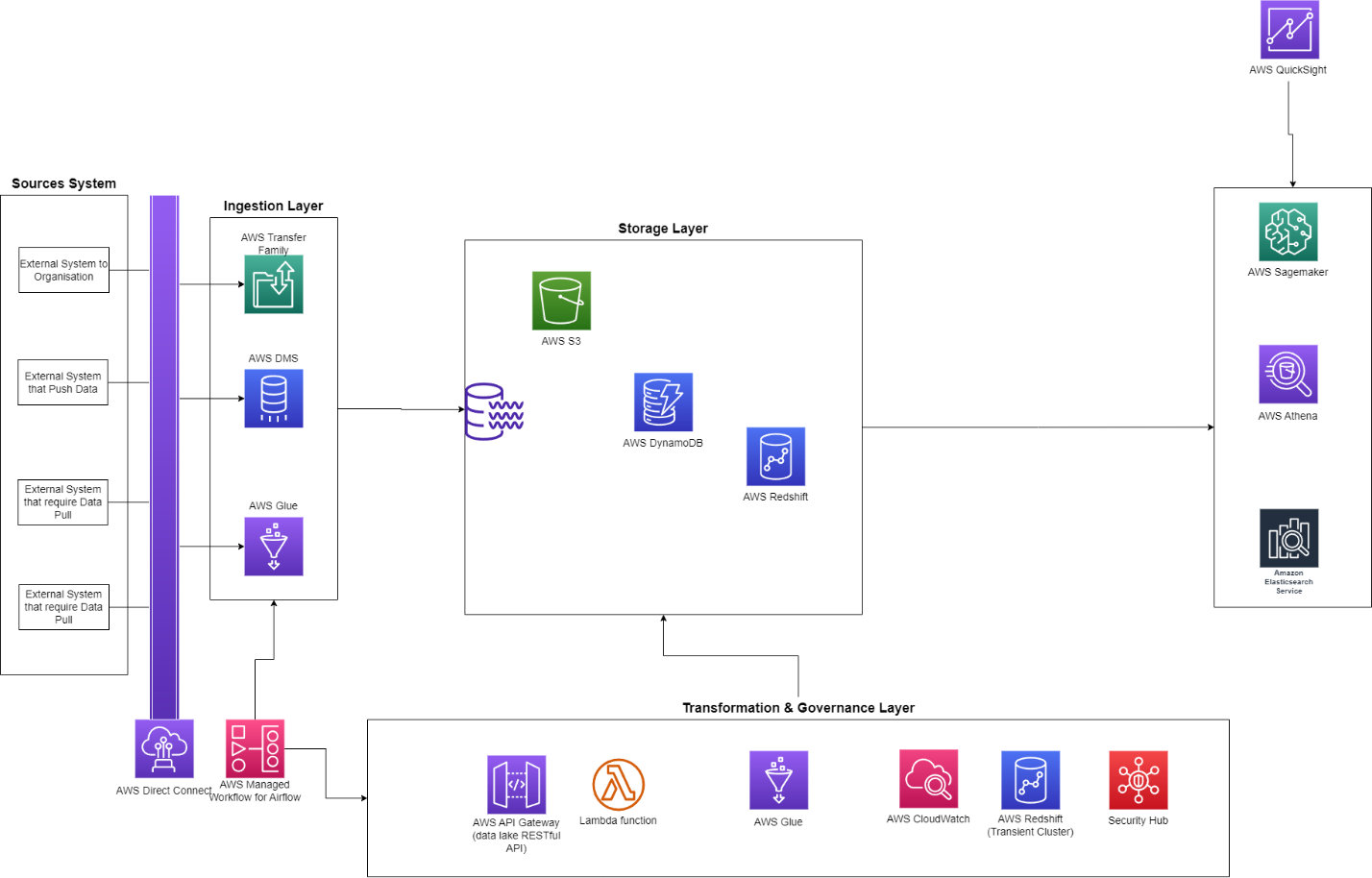
<What data is missing in the problem statement, and you made assumptions about it to create the architecture?>

<Describes any potential risks that may be created now or in future based on these assumptions>

[You may not use this example in your final solution] e.g.:

1. Hadoop cluster will use Linux operating system
2. Data Lake will not support X, Y, Z

# Data Lake Architecture for Medical Data Processing Company



# Design Considerations and Rationale <at least 3 pages>

## Ingestion Layer

<How do you plan to ingest different types of data?>

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

<What tools would be used? Why? >

<How would the ingestion layer design scale?>

<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?>

## Storage Layer

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

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

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

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

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

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

<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?>

## Processing Layer

<How do you plan to process the data?>

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

<How do you enable ad-hoc querying capabilities?>  
<What different tools are involved for processing?>

<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?>

<How does the proposed architecture scale with respect to processing?>

## Serving Layer

<What do you mean by serving layer?>

<What type of data do you plan to store here?>

<How would the data in the serving layer be used?>

# 8. Conclusion <approx 2-5 lines>

<Conclude the contents of the document. Provide recommendations on next steps if any.>

# 9. References <If any>

<Provide links of any external documentation, wiki, blogs that you used to complete your research to put this solution together>