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 |
| 06/04/2020 | FirstName LastName | 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 |
| Pouya Ataei | Data Architect | Medical Data Processing | Pouya.ataei.7@email.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 this document is to delineate the new data architecture designed for DataProcessingCo. This document contains the purpose, the requirements, the design, the assumptions, and the rationale behind the design. I’m creating this document to effectively communicate the properties of this new data design and justify its usage to solve some of the current problems.   
  
This is document is targeted for enterprise architect, data architects, data engineers, software engineers and technical directors. In-scope items of this document are purpose, requirements, design, assumptions and the rationale. Anything other than these, are out-of-scope items.

# Requirements

Some of the important requirements of the new data lake architecture is to improve up-time, reduce latency, increase system reliability, introduce horizontal scaling, and having a metadata-driven design. Current data architecture is inflexible and overly centralized using vertically scaled relational databases at its core. This architecture has been challenging to scale and maintain.

# 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

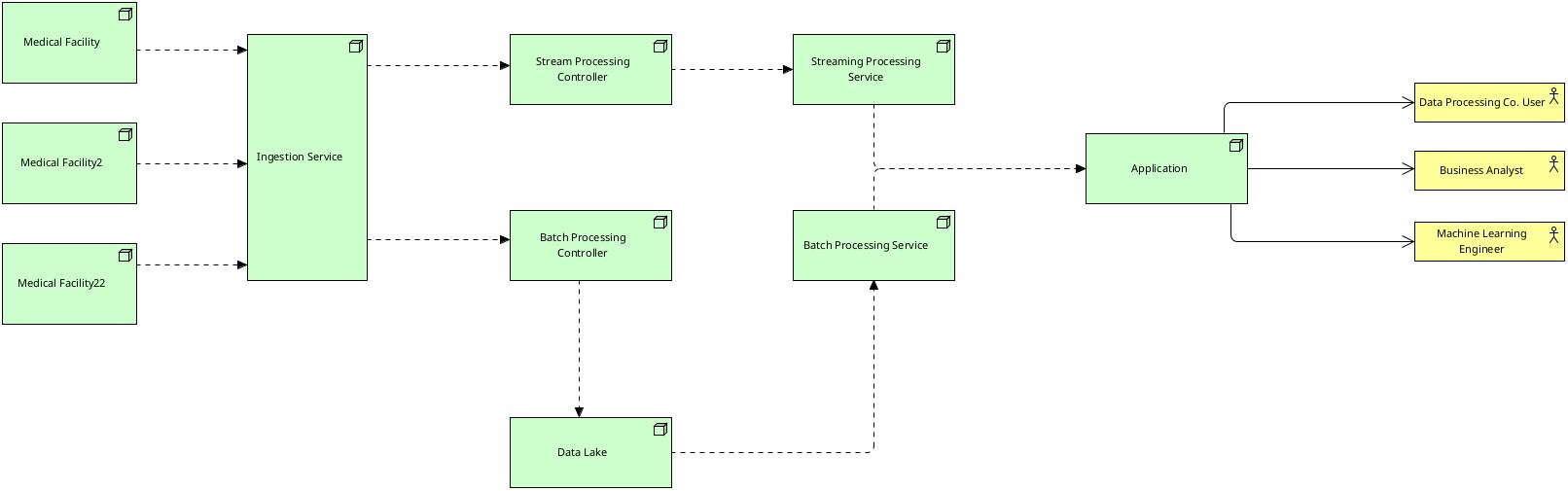
# Data Lake Architecture design principles

* Scalability: The architecture should be able to accommodate growth in the volume, variety, and velocity of data.
* Flexibility: The architecture should allow for changes in business requirements and technology advancements.
* Maintainability: The architecture should be easy to maintain, with clear documentation and modular components.
* Integration: The architecture should support seamless integration with other systems and technologies.
* Data Quality: The architecture should ensure the accuracy, completeness, consistency, and reliability of data.
* Security: The architecture should provide secure access to data and protect sensitive information.

# Assumptions

* There is a need for both batch and stream processing and therefore there should be separate services associated to these two different processing models.
* Data should be stored in data lake for metadata management and bulk batch processing.
* Applications can retrieve their data from stream processing and batch processing services.
* There should be an ingress that retrieves data from the customers.
* Stream processing and batch processing layer should be separated.
* The ingress could be just a simple Nginx server that handles different requests and forwards them to different services based on the nature of the request.

# 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>