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 | Nawaf Alomeir | 0.1 | Initial draft |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
| Nawaf Alomeir | 1.0 | Udacity Reviewer  Enterprise Data Lake Architect | 5/14/2024 |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
| Nawaf Alomeir | Data Architect | Medical Data Processing | 441050905@std.psau.edu.sa |

DESIGN Data Lake

Purpose

Summary

Medical Data Processing Company’s current data armature ca n’t keep up with the growth. As the volume of

data continues to grow, the being single knot SQL Garçon isn't suitable to gauge . The SQL Garçon has come a

single point of failure, hosting critical client data. The CTO is looking for a data lake result. This is the

comprehensive design document with the proposed data lake armature.

Document figure

• Data Lake Conditions

• Data Lake Architecture design principles

• hypotheticals

• Data Lake Architecture Offer

• Design Considerations and explanation

• Conclusion

Target followership

Data lake stakeholders including

• Company leadership

• Data masterminds

• Data scientists

• External mates

In compass and Out of compass particulars

• In compass the conditions, the hypotheticals, the design of the data armature

• Out of compass perpetration of the data armature, data governance, machine literacy

Data Lake Conditions

Summary of conditions for Data Lake

• Design a system with high vacuity, trustability, and resiliency

• Scale fluently to keep up with the growth

• Break down data silos and maintain one source of verity

• Integrate flexibly with ML fabrics, reports and dashboards

Being Technical Environment

• 1 Master SQL DB Garçon

• 1 Stage SQL DB Garçon

• 64 core vCPU

• 512 GB RAM

• 12 TB fragment space( 70 full,

TB)

• 70 ETL jobs running to manage over 100 tables

• 3 other lower waiters for Data Ingestion( FTP Garçon, data and API excerpt agents)

• Series of web and operation waiters( 32 GB RAM Each, 16 core vCPU)

Current Data Volume

• Data coming from over 8K installations

• 99 zip lines size ranges from 20 KB to1.5 MB

• Edge cases some large zip lines are as large as 40 MB

• Each zip lines when unbolted will give either CSV, TXT, XML records

• In case of XML zip lines, each zip train can contain anywhere from 20- 300 individual XML lines, each XML train with

one record

• Average zip lines per day 77,000

• Average data lines per day

• Average zip lines per hour 3500

• Average data lines per hour 700,000

• Data Volume Growth rate 15- 20 YoY

Business Conditions

• Ameliorate uptime of overall system

• Reduce quiescence of SQL queries and reports

• System should be dependable and fault tolerant

• Architecture should gauge as data volume and haste increases

• Ameliorate business dexterity and speed of invention through robotization and capability to experiment with new

fabrics

• Embrace open source tools, avoid personal results which can lead to seller cinch- in

• Metadata driven design- a set of common scripts should be used to reuse different types of incoming data

sets rather than erecting custom scripts to reuse each type of data source.

• Centrally store all of the enterprise data and enable easy access

Specialized Conditions

• Capability to reuse incoming lines on the cover( rather of nocturnal batch loads moment)

• Separate the metadata, data and cipher/ processing layers

• Capability to keep unlimited literal data

• Capability to gauge up processing speed with increase in data volume

• System should sustain small number of individual knot failures without any time-out

• Capability to perform change data prisoner( CDC), UPSERT support on a certain number of tables

• Capability to drive multiple use cases from same dataset, without the need to move the data or prize the data

• Capability to integrate with different ML fabrics similar as TensorFlow

• Capability to produce dashboards using tools similar as PowerBI, Tableau, or Microstrategy

• induce daily, daily, nocturnal reports using scripts or SQL

• Ad- hoc data analytics, interactive querying capability using SQL

Data Lake Architecture Design Principles

Use event sourcing to insure data traceability and thickness

In a data lake armature where cipher and storehouse are separated, event sourcing should be used and an

inflexible log of all incoming events should be maintained on object storehouse. Event sourcing enables you to

retrace the way to learn about the exact metamorphosis applied on the raw data, down to the event position. If there

was an issue in your ETL law, you can fluently fix it and run the new law on the inflexible original data.

Subcaste data lake according to stoner’s chops

In a data lake, we've the possibility to store multiple clones of the data for different use cases and consumers.

By automating the ETL channels that ingest the raw data and perform the applicable metamorphoses per use case

we can help the data engineering tailback that might form if we calculate on rendering- grounded ETL fabrics similar

as Apache Spark.

Keep the armature open

To produce an open armature, you should

• Store the data in open formats like Avro and Parquet which are standard, well- known and accessible by

different tools.

• Retain literal data in object storehouse like Amazon S3.

• Use a central meta- data depository similar as AWS Cohere. This will allow you to polarize and manage all your

meta- data in a single position, reducing functional costs in structure, IT coffers and engineering hours.

Plan for performance

To insure high performance when querying data, you need to apply storehouse stylish practices to make data extensively

available

• You want every train stored to contain the metadata demanded in order to understand the data structure.

• Use columnar train formats similar as Apache Parquet and ORC.

• Keep your data in optimal train sizes. A Hot/ Cold armature is recommended hot – small lines for good

newness; cold wave – incorporating small lines into bigger lines for better performance.

• make an effective partitioning strategy to insure queries run optimally by only reacquiring the applicable data demanded

in order to answer a specific logical question.

hypotheticals

• Timeline for transitioning to the data lake ASAP.

• pall is preferred over on- premise structure. An on- premise data lake imposes challenges. Companies

must make their own data channels, pay the ongoing operation and functional costs in addition to the original

investment on waiters and storehouse outfit, and manually add and configure their waiters to gauge a data

lake to feed to further druggies or adding data volume.

• 100 of the data is migrated to the pall.

Data Lake Architecture

I ’ll influence the AWS Data Lake result. The overall services can be grouped into the following five orders

• Centralized storehouse that can gauge as per the business needs.

• Managed ingestion to onboard data from colorful sources and any format.

• Processing and assaying at big data scale in colorful programming languages.

• Serving the reused data to consumer operations.

• Governing and securing your data packages.

Design Considerations and Rationale

Store

This data lake armature is grounded on Amazon S3 as the primary patient data store. We can separate

cipher and storehouse, and resize on demand with no data loss. It's also largely durable and low- cost with several

options to connect. Spreading requests across numerous connections is a common design pattern to horizontally

scale performance. Amazon S3 does not have any limits for the number of connections made to the pail. Data

will be stored in the native formats upon ingestion. After ETL, data can be stored in open formats similar as Apache

Parquet and ORC that are standard, well- known and accessible by different tools. The metadata can be peopled

by AWS Cohere, the web interface or via the API, and stored in DynamoDB. Data can be participated with multiple data

lakes through the S3 pails. Data can be replicated in different regions for back- over and recovery.

Because SQL- grounded access is demanded, Redshift is added to the storehouse subcaste. Redshift is a fast and completely managed

petabyte- scale data storehouse that costs lower than$ 1,000 per terabyte per time. The Redshift cluster can be

resized to change the knot type, number of bumps, or both.

I considered the pall storehouse from Microsoft and Google. But Amazon S3 is a better choice in terms of the cost

and performance.

Ingest

Amazon has several tools that can ingest data to S3 and Redshift.

• Direct Connect establishes private connectivity between AWS and the enterprise data center and provides an

easy way to move data lines from the operations to S3. AWS Direct Connect makes it easy to gauge your

connection to meet your requirements. AWS Direct Connect provides 1 Gbps and 10 Gbps connections, and you can

fluently provision multiple connections if you need further capacity.

• Snowball significances hundreds of terabytes of data snappily into AWS using Amazon- handed secure appliances for

secure transport. Multiple bias can be used in resemblant or clustered together to transfer petabytes of data into

or out of AWS.

• Kinesis and Kinesis Firehose enable the structure of custom operations that process or dissect streaming data.

Kinesis Data Aqueducts now supports spanning up to 10,000 MB/ s outturn with a single API call.

I considered open- source tools including Apache Sqoop, Flume, Kafka and Nifi but decided to use managed

services Amazon handed.

• Apache Sqoop is a tool designed for efficiently transferring data between structured,semi-structured, and

unshaped data sources. Sqoop works on top of Hadoop HDFS and once a Sqoop job is submitted, it gets

converted to a MapReduce job.

• Flume is a distributed, dependable, and available service for efficiently collecting, adding up , and moving large

quantities of log data. It has a simple and flexible armature grounded on streaming data flows. Flume, just like

Scoop Leverages Hadoop. All the Flume jobs are internally converted into MapReduce. Apache Flume

armature consists of three major factors, sources, channels, and sinks. A Flume source consumes

events delivered to it by an external source like a web garçon. The external source sends events to the flume in a

format that's honored by the target Flume source. When a flume source receives an event, it stores it into

one or further channels, which keep the event until it's consumed by a flume Gomorrah.

• Apache Kafka, is an open source distributed event streaming platform. Kafka is used by thousands of

companies, for high performance data channels and streaming analytics. Kafka has a veritably different set of use

cases. Kafka is horizontally scalable, fault-tolerant, fast frame, for erecting real time data channels.

• Apache NiFi is an intertwined data logistics platform for automating the movement of data between distant

systems. It provides real- time control that makes it easy to manage the movement of data between any destination.

Process

For data processing and analysis, the services from AWS will be used. They've erected- in scalability.

• Lambda runs law without provisioning. It can be touched off upon appearance of data in S3 and for any streaming

sources from Kinesis.

• Cohere is a service that enables ETL and makes it easy to transfigure and move data from S3 to the consumers. It

is integrated with S3, Redshift, and other JDBC biddable data sources and bus- suggests schemas and

metamorphoses, which ameliorate inventor productivity. You can also view and edit the law it generates in

popular languages similar as Python and Spark, with the capability to partake the law with your peers. Cohere schedules

the ETL jobs and bus- vittles and scales the structure grounded on the job conditions.

• EMR provides a distributed cipher frame that makes it an easy, presto, and cost-effective way to reuse

data on S3 at scale and on demand. AWS now provides options for spot cases that are offered at lower

cost. They're stylish used for operation tests and use cases that do not have hard SLA’s.

• Athena is a query service that makes it easy to dissect data directly from lines in S3 using standard SQL

statements. Athena isserver-less, which makes it really stand out since there's no fresh structure to be

provisioned. Athena can be used as the ad- hoc query tool.

• QuickSight is presto, pall- powered business intelligence( BI) service.

• Machine literacy provides visualization tools and wizards for creating machine literacy models and executing

them on big data.

I considered Apache MapReduce, Pig, Hive and Spark as the processing tool, but decided to use managed

services Amazon handed.

• MapReduce symphonies the processing by marshaling the distributor waiters, running colorful tasks in parallel.

MapReduce is a programming model and an associated perpetration for processing and generating big

datasets with a resemblant distributed algorithms on a cluster. MapReduce also manages all the communication

and the data transfers between the colorful corridor of the system, furnishing redundancy and fault forbearance. A

MapReduce program executes in three stages, videlicet maps stage, equivocation and kind stage, and the reduce

stage. In chart stage, mapper's job is to reuse the input data. Generally, the input data is in a form of a train or a

directory and is stored on Hadoop train system. The input train is passed to the mapper function line by line. The

mapper processes the data and produce several small gobbets of the data. The affair of the chart stage is

reused using equivocation and kind stage. Where, the data is scuffled across the network of bumps and sorted

grounded on designated key. The affair of equivocation and kind phase is used by the reducer phase. In the reducer

stage, data is added up and also the final affair is stored on HDFS. The MapReduce frame manages all

the details of data passing, similar as issuing task, vindicating task completion, and copying the data around the

cluster between the bumps. This really simplifies the data processing across a distributed system.

• Apache Pig is simple to use, yet veritably important frame. Pig is a platform for assaying large data sets that

consists of a high position language for expressing data analysis programs. Apache Pig works on top of HDFS,

coupled with structure for assessing this program similar as MapReduce. Pig offers ease of programming.

Complex task comprised of multiple interrelated data metamorphoses are veritably easy to write, understand, and

maintain. For illustration, a simple word count program that can take 60 to 70 lines of Java MapReduce law can

be expressed with lower than ten lines of Apache Pig law. Pig frame also optimize the prosecution

automatically, allowing you to concentrate on the semantics rather than the effectiveness. Pig frame is veritably flexible

and offers great deal ofextensibility.However, and if the erected- in Pig functions are not

If you're doing atransformation.enough, you can produce your own functions and do spatial purpose processing. These functions are called stoner

defined functions.

• Apache Hive is a data storehouse software which allows you to use SQL language to read, write, and manage

large datasets abiding on Hadoop distributed storehouse. Hive was firstly developed by Facebook, but latterly it

was open sourced. Hive is a veritably popular tool and used by some of the largest tech companies around the

globe. Hive gives a SQL- suchlike interface to query data stored in colorful databases and train systems that integrate

with Hadoop. Hive comes with command line tool and JDBC motorist to connect the druggies to Hive

programmatically. Hive provides necessary SQL abstraction to integrate SQL like queries, also known as Hive

query language into the underpinning Java without the need of actually enforcing the queries in a low position

Java MapReduce API. principally, to interact data stored on Hadoop, you don't need to write really long and

complicated MapReduce law in Java. You can interact with the same data by writing SQL queries and Apache

Hive frame will convert the SQL queries into MapReduce law behind the scenes.

• Spark provides presto unified data processing capabilities on a large quantum of data stored on Hadoop. Spark is a

unified analytics machine for large- scale data processing. Before spark, inventors will have to use different set of

tools depending on the usecase.However, they will use Pig or Hive, If they had batch data processing. If they wanted to

do real- time data processing, also they will use Apache Storm. still, Spark brings all of them together, and

now you can use Spark to do real- time as well as batch data recycling use cases under one single unified

frame. Spark can run workloads 100 times faster than MapReduce. You can use it interactively from the

programming language of your choice, similar as Scala, Python, R, and SQL.

Serve

The core task of the serving subcaste is to expose the views created by the process & dissect subcaste for querying by

other systems or druggies. The serving subcaste consists of data that can be readily served to consumer operations.

Hence this is substantially the reused data.

AWS offers a comprehensive suite of pall-native results to allow consumers to query and dissect the data

directly. Common analytics approaches are covered, including Amazon Redshift for data warehousing; Amazon

Athena for SQL querying on demand; Amazon EMR for running popular open source fabrics similar as

Hadoop, Spark, Presto, Flink, and HBase; Amazon QuickSight for business intelligence; and Amazon

Elasticsearch Service for logs and textbook. There's no need to resettle data to different operating surroundings,

helping to avoid the coexisting outflow, costs, trouble, or detainments. Amazon EMR, Amazon Redshift Spectrum,

and Amazon Athena allow consumers to query data directly in Amazon S3 and, with AWS Cohere, they all can partake

the same data roster. also, artificial intelligence( AI) and machine literacy are getting decreasingly

popular tools for erecting smart operations similar as prophetic analytics and deep literacy. To make it more

accessible, Amazon Machine Learning objectifications down from the algorithms with wizards, APIs, and guidance. AI

services include Amazon Polly for textbook- to- speech, Amazon Lex for natural language processing and conversational

bots, and Amazon Rekognition for image identification and bracket. Any of the labors can be served to the

consumer operations.

Govern & Secure

Governance is crucial in erecting a managed data lake. Below are the the key services used for governance

• stoner and access operation AWS Data Lake result provides a web interface to manage druggies for the data

lake. As a data lake director, you can decide which stoner gets access to the data lake and at what position

( member or director). You can also grant API access to specific druggies.

• Data roster As data is used in different platforms, ETL( Excerpt, transfigure, cargo) is an important function to

insure that it's moved and understood duly. AWS Cohere is an ETL machine that can be used to understand

data sources, prepare data, and load it reliably to data stores. AWS Cohere discovers the data and stores the

associated metadata(e.g. table description and schema) in the AWS Cohere Data roster. Once entered, the

data is incontinently searchable, can be queried, and is available for ETL.

• Data roster stoner interface The data roster is searchable using the web operation. The roster can be

peopled by the web interface or via the API with information about the colorful packages for the data lake, and

this information is stored in DynamoDB. Once the datasets are registered, they're automatically listed to

Elasticsearch and are searchable by the web interface.

• Security and compliance AWS has a deep suite of security immolations like Amazon Macie, a security service that

uses machine literacy to automatically discover, classify, and cover sensitive data. The data center and

network armature were erected to meet the conditions of the most security-sensitive associations. AWS also

laboriously manages dozens of compliance programs in its structure, helping associations to meet compliance

norms similar as PCI DSS, HIPAA, and FedRAMP. Below are the stylish practices recommended by AWS to

secure the data

1) Classify the data

1) Identify the data within your workload

2) Automate identification and bracket

3) Define data lifecycle operation

2) cover data in rest

1) apply secure crucial operation

2) apply encryption at rest

3) Automate data at rest protection

4) apply access control

3) cover data in conveyance

1) apply secure key and instrument operation

2) apply encryption in conveyance

3) Automate discovery of unintended data access

4) Authenticate network communication

8. Conclusion

To epitomize, below is the armature of the proposed data lake

• A data lake on AWS leverages S3 for secure, cost-effective, durable, and scalable storehouse. Amazon S3 also

offers an expansive set of features to give strong security for the data lake, including access controls and

programs, data transfer over SSL, encryption for data at rest and in stir, logging and covering, and more.

Redshift is added to the storehouse subcaste for SQL- grounded access and costs lower than$ 1,000 per terabyte per time.

• Data can be snappily and fluently ingested into Amazon S3 from the SQL Garçon, FTP garçon and APIs by Direct

Connect, AWS Snowball, and Amazon Kinesis.

• For processing and assaying the data stored in Amazon S3, AWS provides fast access to flexible and low- cost

services, like Amazon EMR, Amazon Redshift with Redshift Spectrum, Amazon Athena, and Amazon AI

services, so any logical result can be fleetly gauged to power any big data operations, meet demand, and

ameliorate invention.

• For governing and securing the data, AWS Cohere, Amazon DynamoDB, and Amazon ElasticSearch can be

abused to roster and indicator the data in Amazon S3. Using AWS Lambda functions that are directly touched off

by Amazon S3 in response to events similar as new data being uploaded, the roster can be fluently kept up- todate. With Amazon API Gateway, API can be created that acts as a “ frontal door ” for operations to pierce data

snappily and securely by authorizing access via AWS Identity and Access Management( IAM) and Amazon

Cognito.

previous to the data lake perpetration, the coming step is to jut down the use cases. Defining a many most poignant

use cases for the data lake should take precedence over deciding the technology involved. The defined use cases will

help to showcase some immediate returns and business impact of the data lake, which will be crucial to maintaining

design support from the advanced up the chain of command, and design instigation.9. References:

1. Amazon Direct Connect
2. Amazon Snowball
3. Apache Kafka: Ingest real-tome streaming data to HDFS
4. Apache Nifi: Supports powerful and scalable directed graphs of data routing, transformation, and system
5. mediation logic
6. AWS (S3) Simple Storage Service
7. Amazon Redshift
8. Blob Storage
9. Data Lake Storage
10. GCP Cloud Storage
11. Amazon EMR (Elastic Map Reduce): Easily run and scale Apache Spark, Hive, Presto, and other big data
12. frameworks
13. Amazon Athena
14. Amazon Machine Learning
15. Amazon QuickSight
16. Amazon SageMaker
17. Amazon Lambda
18. Apache MapReduce
19. Apache Pig
20. Amazon Glue
21. Amazon DynamoDB
22. Amazon Elasticsearch
23. Amazon API Gateway