MANIPAL
ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES
(A Constituent unit of MAHE, Manipal)

Web Interface to ingest heterogeneous data into Datalake

Reg. Number	Name	Branch
211047015	Natasha Avryl Lasrado	Cloud Computing
211047016	Ganesh Prasad UT	Cloud Computing

Under the guidance of

Prof. Sreepathy H V
Assistant Professor – Senior Scale,
Manipal School of Information Sciences,
MAHE, MANIPAL

12/05/2022

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

DECLARATION

We declare that this mini project, submitted for the evaluation of course work of Mini Project to Manipal School of Information Sciences, is our original work conducted under the supervision of our guide Sreepathy H V and the panel members, Dr. Ravindra B. V., Mr. Samarendranath Bhattacharya.

Natasha Avryl Lasrado
211047015

Ganesh Prasad UT
211047016

ABSTRACT

The process of data ingestion is done through the web interface. However, because sources can be different database systems or any local repository, structured, semi-structured and unstructured data complicate the ingestion procedure. It is usually not enough to just store everything. Usually the unstructured data is stored in databases in cloud platform and in some repository. The user needs to ingest data from specific git location. It needs to be stored in such a way that enables users to quickly access and manipulate it. Web dashboard is generated to obtain data from various sources and a complete information about the owner who ingested the data is being obtained in the interface. Datalake is being created using object storage tool called MinIo.

Keywords: Web Interface, Dashboard, Datalake, Ingestion, Repository, Structured, Semi Structured, Unstructured data.

Contents

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

A Data Lake is a storage repository that can store large amount of structured, semi-structured, and unstructured data. It is a place to store every type of data in its native format with no fixed limits on account size or file. It offers high data quantity to increase analytic performance and native integration. A Web Interface is developed to ingest the heterogenous data like structured, unstructured and semistructured data from various sources into Datalake using StreamLit Python framework. Web dashboard is generated to obtain data from various sources and a complete information about the owner who ingested the data is being obtained in the interface. Data from different databases like MongoDb for storing the documents, user specified git repository and cloud platforms for storing images are ingested to the minio object storage.

1.1 Data Lake

Data Lake is like a large container which is very similar to real lake and rivers. Just like in a lake you have multiple tributaries coming in, a data lake has structured data, unstructured data, machine to machine, logs flowing through in real-time.

The Data Lake democratizes data and is a cost-effective way to store all data of an organization for later processing. Research Analyst can focus on finding meaning patterns in data and not data itself. Unlike a hierarchal Data Warehouse where data is stored in Files and Folder, Datalake has a flat architecture.

Every data elements in a Data Lake is given a unique identifier and tagged with a set of metadata information. A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale.

Data is stored as-is, without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions.

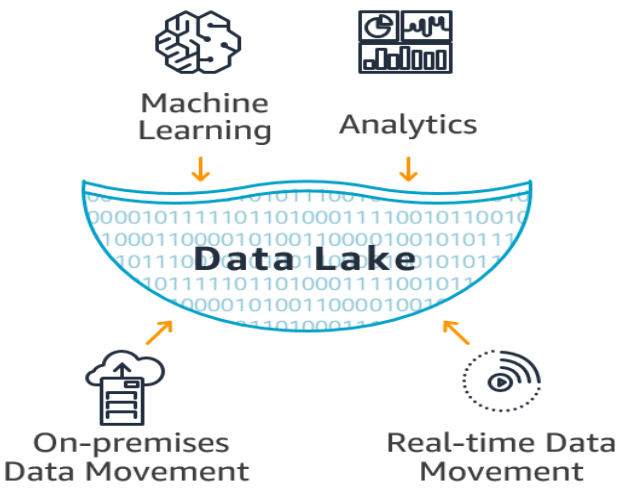


Figure 1: Representation of Data Lake

1.2 MinIO

MinIO is a High Performance Object Storage released under GNU Affero General Public License v3.0. It is API compatible with Amazon S3 cloud storage service. It can handle unstructured data such as photos, videos, log files, backups, and container images with (currently) the maximum supported object size of 5TB.

MinIO Server

MinIO cloud storage server is designed to be minimal and scalable. It is light enough to be bundled along with the application stack, similar to NodeJS, and Redis. Designed for high performance, MinIO offers a suite of features that are specific to large enterprise deployments, these include erasure coding, bitrot protection, encryption/WORM, identity management, continuous replication, global federation, and multi-cloud deployments via gateway mode. MinIO server is hardware agnostic, it can be installed on physical or virtual machines or launched as Docker containers and deployed on container orchestration platforms like Kubernetes.

MinIO Client

MinIO Client SDK provides an API to access any Amazon S3 compatible object storage server. Language bindings are available for Go, Java, Python, JavaScript, Haskell, and languages hosted on top of the .NET Framework.

CHAPTER 2

2. ARCHITECTURE

2.1 Data Lake Architecture

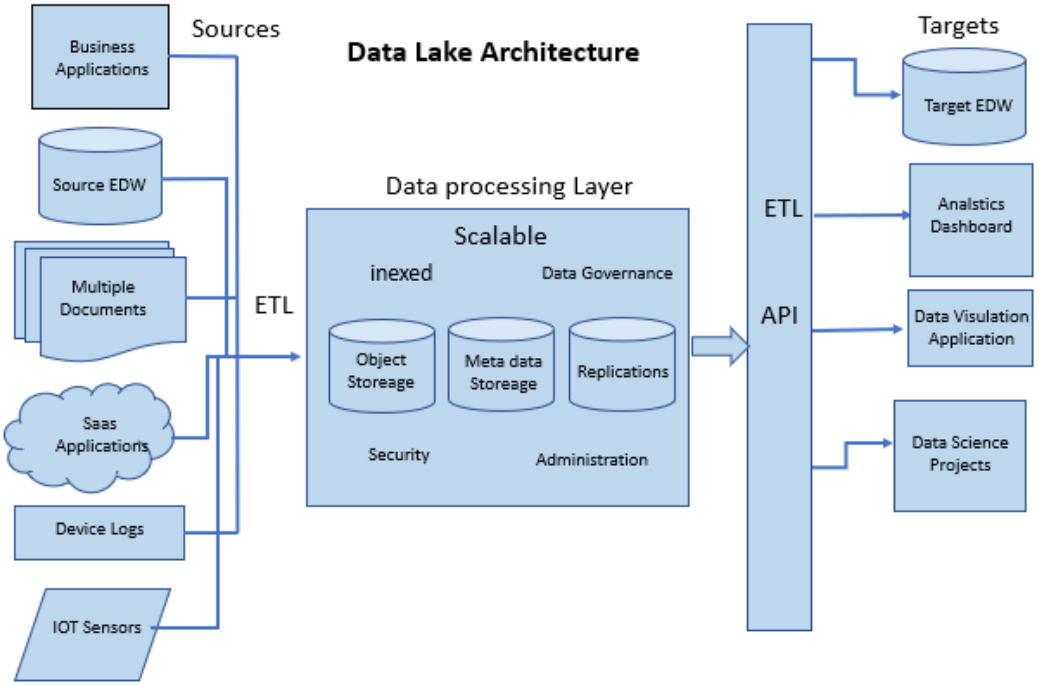


Figure. 2. Data Lake Architecture

The figure shows the architecture of a Business Data Lake. The lower levels represent data that is mostly at rest while the upper levels show real-time transactional data. This data flow through the system with no or little latency. Following are important tiers in Data Lake Architecture:

Ingestion Tier: The tiers on the left side depict the data sources. The data could be loaded into the data lake in batches or in real-time

Insights Tier: The tiers on the right represent the research side where insights from the system are used. SQL, NoSQL queries, or even excel could be used for data analysis.

Distillation tier takes data from the storage tire and converts it to structured data for easier analysis.

Processing tier run analytical algorithms and users queries with varying real time, interactive, batch to generate structured data for easier analysis.

Unified operations tier governs system management and monitoring. It includes auditing and proficiency management, data management, workflow management.

2.2 MinIO Architecture

MinIO's enterprise class features represent the standard in the object storage space. From the AWS S3 API to S3 Select and our implementations of inline erasure coding and security, our code is widely admired and frequently copied by some of the biggest names in technology and business.

MinIO protects data with per-object inline erasure coding written in assembly code to deliver the highest possible performance.

MinIO's optimized implementation of the Highway Hash algorithm ensures that it will never read corrupted data - it captures and heals corrupted objects on the fly. Integrity is ensured from end to end by computing a hash on READ and verifying it on WRITE from the application, across the network and to the memory/drive. The implementation is designed for speed and can achieve hashing speeds over 10 GB/sec on a single core on Intel CPUs.

MinIO supports the most advanced standards in identity management, integrating with the OpenID connect compatible providers as well as key external IDP vendors. That means that access is centralized and passwords are temporary and rotated, not stored in config files and databases. Furthermore, access policies are fine grained and highly configurable, which means that supporting multi-tenant and multi-instance deployments become simple.

MinIO's continuous replication is designed for large scale, cross data center deployments. By leveraging Lambda compute notifications and object metadata it can compute the delta efficiently and quickly. Lambda notifications ensure that changes are propagated immediately as opposed to traditional batch mode.

Continuous replication means that data loss will be kept to a bare minimum should a failure occur - even in the face of highly dynamic datasets. Finally, like all that MinIO does, continuous replication is multi-vendor, meaning that your backup location can be anything from NAS to the public cloud.

CHAPTER 3

3. DESIGN

1. Download the executable from MinIO server and client using **wget** and make these files executable using the following commands.

   $wget https://dl.min.io/server/minio/release/linux-amd64/minio

   $ chmod +x minio

   $ wget https://dl.min.io/client/mc/release/linux-amd64/mc

   $ **chmod +x** mc

2. For MinIO to work, a local directory is required to store the data. Create a directory in a local file system, and start the MinIO server pointing to that local directory.

   $ mkdir minio_storage

   $ ./minio server /home/demo/minio_storage

3. Noting down the Endpoint IP address, AccessKey, and SecretKey in order to access the MinIO through a web browser.

4. Create an alias called “myminio” for our MinIO instance by executing the following command in a new terminal window.

   $ ./mc config host add myminio http://127.0.0.1:9000 minioadmin minioadmin

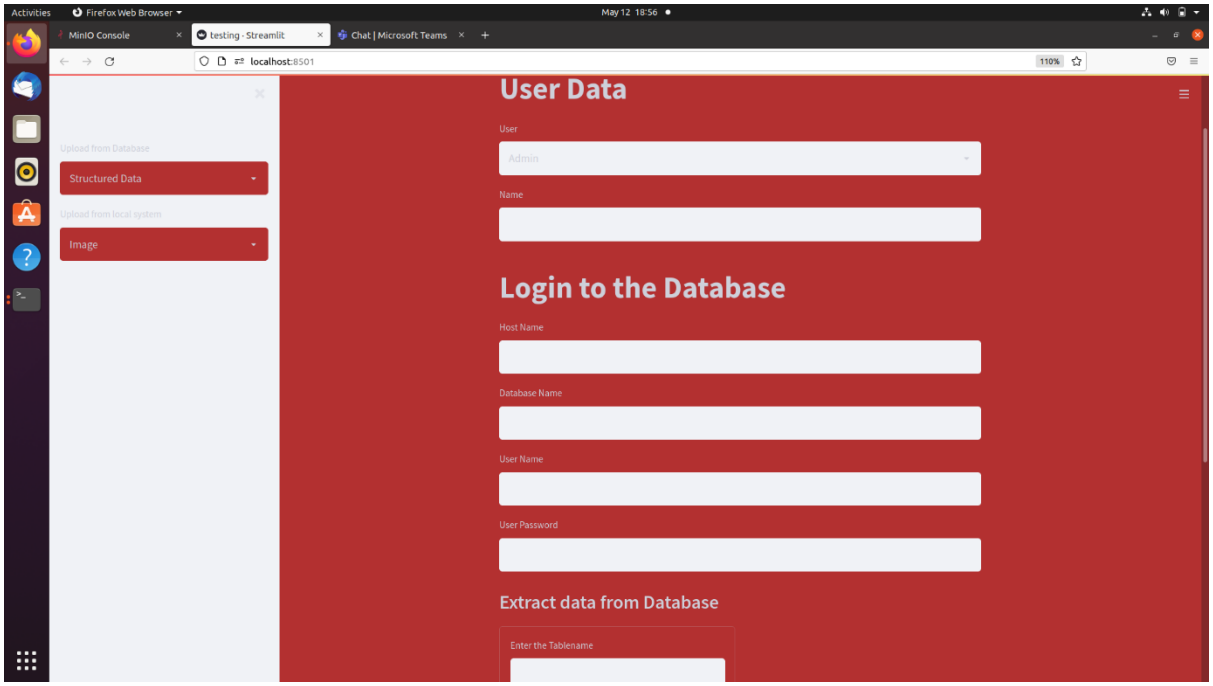
5. Create first S3 bucket called “**datalake**”

   $ ./mc mb myminio/datalake

STREAMIT:

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc. With Streamlit, no callbacks are needed since widgets are treated as variables. Data caching simplifies and speeds up computation pipelines. Streamlit watches for changes on updates of the linked Git repository and the application will be deployed automatically in the shared link.

Streamlit makes it easy to visualize, mutate and share data. The API reference is organized by activity type, like displaying data or optimizing performance.



CHAPTER 4

4. FUTURE WORK

A Web Interface is developed to ingest the heterogenous data like structured, unstructured and semistructured data from various sources into Datalake using StreamLit Python framework. Different databases are being setup from which the data could be ingested and stored as an object storage. Data from user specified git repository is also needed to be uploaded in minio. The owner, context and privacy of the data is identified in the web dashboard.

REFERENCES

1] Hassan Mehmood, Ekaterina Gilman, Marta Cortes, Panos Kostakos, Andrew Byrne, Katerina Valta, Stavros Tekes, Jukka Riekki , “Implementing Big Data Lake for Heterogeneous Data Sources”, 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW).

2] Aleksandar Tunjić, Zagreb, Croatia ,“The Automation of the Data Lake Ingestion Process from Various Sources”, MIPRO 2019, May 20-24, 2019, Opatija Croatia.

3] https://www.guru99.com/data-lake-architecture.html

4] https://min.io/product/overview