USN NUMBER: 1RVU23CSE153
NAME:Ekshu DP

Ex No: 5 Date: 19/08/25	Building an Open-Source Data Lake

Objective:

The primary objective of this lab was to establish a fully functional, open-source Data Lake environment using Docker Compose. This involved successfully integrating three core services: **MinIO** (for S3-compatible object storage), **Hive Metastore** (for centralized metadata management), and **Trino** (for distributed, high-speed querying).

Outcomes:

The deployed solution establishes a three-tiered data lake architecture:

- **Storage Layer (MinIO):** Acts as the secure, durable object store (similar to AWS S3).
- **Metadata Layer (Hive Metastore):** Tracks the schema and physical location of data files stored in MinIO.
- **Compute Layer (Trino):** Serves as the high-performance SQL query engine, reading data from MinIO based on metadata provided by the Hive Metastore.

Materials
This activity focuses on container orchestration for a streaming platform.

A. Software and Platform

Material	Purpose
Operating System	Windows, macOS, or Linux
Docker Desktop	Required to run containers and the Docker Engine.
Docker	Tool to manage multi-container applications (usually

Compose	integrated with Docker Desktop).
Command Line Interface	PowerShell, Terminal, or Git Bash to execute the docker compose up command.

B. Configuration Files

Material	Purpose
docker-compose.yml	The main configuration file defining the Kafka and Zookeeper services, networking, environment variables (like Broker ID and Zookeeper connection), and exposed ports (9092, 2181).
Kafka Image	The specific Docker image used for the Kafka broker (e.g., wurstmeister/kafka or bitnami/kafka).
Zookeeper Image	The specific Docker image used for Zookeeper (e.g., zookeeper:3.6.3).

---

2. Materials for Trino, Hive Metastore, and MinIO Data Lake Activity

This activity establishes an on-premise cloud-native data lake environment.

A. Software and Platform

Material	Purpose
Operating System	Windows, macOS, or Linux

Docker Desktop/Compose	Essential for defining and running the multi-service architecture.
SQL Client	A tool like DBeaver, psql, or the Trino CLI is needed to connect to the Trino Coordinator (port 8080) for querying and configuration.
Web Browser	Required to access the MinIO web console (typically port 9001) for storage management.

B. Configuration Files

Material	Purpose
docker-compose.yml	Defines all four required services: **MinIO**, **PostgreSQL** (to serve as the HMS backend database), **Hive Metastore**, and **Trino Coordinator/Worker**.
MinIO Configuration	Defines the access key, secret key, and volume mapping for persistent object storage.
Hive Metastore Configuration	Defines the JDBC connection string to link the Metastore to the PostgreSQL database.
Trino Catalog Files	Configuration files (e.g., hive.properties in Trino's etc/catalog directory) that tell Trino:

Lab Procedure

1. Deployment via Docker Compose

The complete environment was deployed using a single Docker Compose configuration file that defined all service dependencies (MinIO, PostgreSQL, Hive Metastore, Trino Coordinator, Trino Worker).

The deployment command used was:

$ docker compose up -d

2. Verification of Service Status

The successful deployment was confirmed by checking the status of the containers using `docker ps`. All core components reported an `Up` status, confirming their readiness.

Service	Role	Typical Port	Status
MinIO	Object Storage	9000 / 9001	Up
Hive Metastore	Metadata Catalog	9083	Up
Trino Coordinator	Query Engine	8080	Up

3. Data Lake Readiness and Connectivity

The final steps involved setting up the data lake for querying:

- **MinIO Initialization:** An initial storage bucket (e.g., `datalake`) was created within MinIO to serve as the root location for all data assets.
- **Trino Configuration:** The Trino Coordinator was configured with a catalog pointing to the Hive Metastore and the MinIO credentials.
- **Connectivity Check:** Test SQL commands were run in Trino, such as `CREATE SCHEMA` and `SHOW TABLES`, to ensure the query engine, catalog, and storage layers were communicating effectively.

Conclusion

This lab successfully demonstrated the ability to provision a complete, production-grade **Data Lake stack** using open-source tools and Docker containerization. The integration of MinIO, Hive Metastore, and Trino provides a powerful, decoupled architecture suitable for modern data processing.

GitHub Link:https://github.com/ekshu05/lab5fde_record-ss