SP-114 Data Lake Implementation

Development Documentation

CS 4850, Section 02, Spring 2026

Professor Perry, 3/8/2026

	Name	Role
Team Leader	Caleb Cox	Document project progress; Coordinate communications and development between team members
Team Member	Bryce Wishart	Documentation, development, programming, and testing.
Advisor/Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.

Caleb Cox
Team Leader

Bryce Wishart
Group Member

Development

One key aspect of this project is the data that we collect and organize through database schema. We decided to use automotive information as a starting point for information gathering, so we sourced data from the Carfax website, pulling data on available cars. Once we gathered enough data, we could transfer the data to the object store.

The data sourcing program was created using Python, with two python files acting as the pipeline. The first one acts as the core logic for gathering data, with the second script inheriting some functions of the first one to add more cars from Carfax, print them to an output JSON file, and detail the progress of each data write.

To store the car listings, we created another script which writes the listing information into a Python dictionary, which has a max size of 512 megabytes. Once reached, a new file is created, and car listings are directed to the new output file. After enough car listings have been gathered, we have another program which connects to the object store using an access key to send all the necessary data into the raw zone of the object store.

Another component to this project is the object store that will function as a repository for all the data gathered, and a place where applications can pull data from for analytics. The MinIO object store is our solution to create a server to store data, as it is an open-source node deployment service. With a free license, our team only has access to a single server node to store data on, which we chose to deploy on a Windows installation. A server was created with several buckets or folders to store data, with incoming data being stored in the data lake raw bucket until they are organized into cleaner schema.

Finally, there is one more application that needs to be created for our database to be accessible as an analytics tool. This last component included the database management system, which we chose to use DuckDB for, and the file formatting system, Apache Parquet. The application uses Python, importing DuckDB and Apache Parquet as libraries to utilize their commands. With DuckDB, the data from our object store can be accessed and categorized by Parquet commands. Once the application identifies the data that fits the schema that we laid out for car information, such as brand, price, or color, then DuckDB can extract a copy of all fitting data and write this information into a document that the user can read and analyze.

Database Connections

The data lake project contains one central object store that serves as a database for several types of data stored in their original formats. Our database was created using the MinIO Alstor server program. Once deployed, this program can be configured to store data on allocated drives using folders known as buckets that can contain multiple versions of the same piece of data. We configured the object store to run from a team member's local computer with an attached external drive that can hold a large quantity of data.

Connecting this central database to our other programs is a key point of the data lake project, and a focal point of development. The Alstor server allows for the connection of an application that is running libraries compatible with Amazon S3, another data storage service. Since our project is a smaller local deployment, the applications for sending and receiving data from the server will just need to have the network address of the store set up for data to transfer to the server.

To allow the address of our store to be visible outside of the local network, we had to create a network tunnel that could funnel requests from a public hostname to the local network address. This was accomplished by using a reverse proxy service known as LocalXpose, which allows for the creation of network tunnels to manage requests.

For sending data out of the object store, our database management program uses DuckDB. This allows us to connect to the object store, since the documentation of DuckDB includes an extension known as httpfs for connecting to S3 compatible object stores like MinIO. By configuring the secret key of the connector to the same key from our object store, the database management application can read and pull data from the server.

Setup and Deployment

The first part of setting up the data lake is starting up the instance of our object store. By configuring the environmental variables beforehand, the only thing needed to start the server is running the minio application in a command line window while pointing to the folder of our implementation. Once the object store is running, then the data from Carfax can be transferred from a computer onto the object store.

After starting the object store, we can run the database management application from DuckDB. This application will pull information from the object store that fulfills the criteria needed and write that information to a new document that the user can read.