**Building a data lake**

***Tools Required:***

* AWS S3 - <https://www.youtube.com/watch?v=e6w9LwZJFIA&t=0s>
* AWS Athena - <https://www.youtube.com/watch?v=K2GfrERtliU&t=0s>
* AWS Glue - <https://www.youtube.com/watch?v=qgWMfNSN9f4&t=0s>
* Python
* Git/Github
* CloudShell - <https://www.youtube.com/watch?v=BmJjQE-l4Jc&t=0s>
* AWS S3: Amazon simple storage service
  + S3 dashboard -> Create bucket -> provide bucket name -> select region - > leave blocks all public access turned on -> object lock disable -> create bucket
  + Upload files in S3 bucket -> drag and drop or aws cli -> set permission on uploaded object -> configure other properties like object storage class -> select standard -> upload
* AWS Athena: Query data in S3 bucket, without putting the data in some type of database.
  + Athena -> create table -> add columns (if crawler is not used) -> can be used bucket to get data faster -> \*\*\*\* -> create table
  + Dashboard -> workgroups -> edit -> query result configuration -> browse s3 -> specify the file path (to output the results) -> save changes -> \*\*\*\*
  + Go over to the database ->
  + Need a separate file location to dump data -> create a folder in s3 -> delete the files created by running the query
  + Workgroup -> specify the new metadata folder
  + Put the data files in a new folder
  + **Might not work for scaling, but only works well for quick insights of your data where the data is stored in s3.**
* AWS Glue: Extract, transform and load service. ETL service. It allows to organize, locate, move and transform all the data sets across the business. Glue is serverless, point glue to all ETL jobs and run. Glue provides crawlers with automatic schema inference, for structure and semi structure data types. Automatically generates scripts to extract, transform and load data to target locations.

***Diving into the project:***

*Architecture:*

S3 buckets (JSON, CSV) -> {Crawler (Apache Spark ETL jobs) -> AWS Glue (Data Catalog, DDL Hive, Metadata)} -> Amazon Athena (SQL Queries, Presto Distributed Engine)

* S3 bucket holding raw data
* Glue will be serverless service for cataloging the data and having the data ready for querying

PickMyVibe – Tuning into user vibes and preferences

**Step 1: Dataset Preparation**

1. **Find a dataset**:
   * Use datasets like MovieLens (movie ratings) or Goodreads (book data).
   * The dataset should include user preferences, ratings, or interaction data.
2. **Upload to S3**:
   * Organize raw data in S3 buckets (e.g., /raw-data).

**Step 2: Data Transformation with AWS Glue**

1. **Set up AWS Glue Crawler**:
   * Crawl the raw data to infer schema and catalog it in AWS Glue Data Catalog.
2. **Create an ETL Job**:
   * Write a Glue script (in Python or Scala) to:
     + Clean data (remove duplicates, handle missing values).
     + Transform data into a format optimized for analysis (e.g., Parquet or ORC).
   * Output processed data to a separate S3 bucket (e.g., /processed-data).

**Step 3: Querying Data with Athena**

1. **Set up Athena**:
   * Configure Athena to query the processed data stored in S3.
   * Use SQL queries to analyze user behavior and preferences.
2. **Build Queries**:
   * Example queries:
     + "Top-rated movies/books by users."
     + "Users who rated similar content highly."
     + "Recommendations for a user based on their interaction history."

**Step 4: Build the Recommendation Algorithm**

1. **Simple Approach**:
   * Use collaborative filtering (e.g., users who liked similar items) or content-based filtering.
   * Use Athena queries to generate recommendation logic.
2. **Advanced Approach**:
   * Export data from Athena and build a machine learning model using Amazon SageMaker or Python libraries like Scikit-Learn or TensorFlow.
   * Train a model like a matrix factorization or neural collaborative filtering model.

**Step 5: Create a Frontend/Visualization**

1. **Option 1:** Integrate the recommendation system into a web app using Flask (you already have experience with Flask!).
2. **Option 2:** Use BI tools like Amazon QuickSight to create a user-facing analytics dashboard.