SCETLS

Team:
Mithali Manjunath - 002879571
Rishikesh Cherodath - 002866379
Thy Tran - 002453818

Use Cases

Data Engineer Configures ETL Pipeline

- **Actor**: Data Engineer
- **Action**: Sets up a new ETL pipeline by defining data sources, transformation steps, and destination in a configuration file.
- **Reaction**: The system validates the configuration file and provides feedback on any issues or missing fields before allowing the job to be run.

Data Analyst Runs Transformation Pipeline in Sandbox Mode

- **Actor**: Data Analyst
- **Action**: Runs a transformation pipeline in sandbox mode to verify outputs on a sample dataset before full deployment.
- **Reaction**: The system executes the pipeline on a small subset of data and returns sample results, allowing the analyst to confirm the transformations without affecting production data.

Business Analyst Checks Data Transformations

- **Actor**: Business Analyst
- **Action**: Views data transformations to check for data consistency, missing values, or anomalies.
- **Reaction**: The system is built in a way to be readable and could highlight things that may require attention.

Data Scientist Requests ETL Job Execution Metrics

- **Actor**: Data Scientist
- **Action**: Requests metrics and a metrics dashboard after running a custom ETL job.
- **Reaction**: The system generates metrics in each transformation stage, including data volume processed and runtime. This would picked up by Prometheus and can be queried

Methodology

- **YAML Parsing:** The multiple YAMLs defining Origin and Destination, Transformations, Quality Checks and other configs would be parsed into an AST and then validated.
- **Source and Destination connectors:** Connectors for the Data Sources and Destination need to be defined.
- **Orchestrator:** An orchestrator will have to be defined to schedule and run jobs based on configuration.
- **Transformation Engine:** A transformation engine will have to be defined that handles transformations from a config. This will also be responsible for running custom transformations written in Scala.
- **Quality Checks:** A Data Quality Check Framework will have to be defined and integrated into every job.
- **Observability and Logging:** Observability support for Prometheus. Logging will have to be standardised.
- **Testing:** Code will require extensive testing to ensure handling of edge cases and the unpredictable nature of dirty data.
- **Sandbox Mode:** Support for a sandbox mode to be able to test configs.

Data Sources

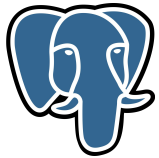
Crime Rates in London

With 7 columns and 13.5m rows

LSOA code : Lower Layer Super Output Area code according to the Office for National Statistics
Borough : Borough of the crime
Major_category : Category of the crime
Minor_category : Subcategory of crime
Values : Number of crimes for the month
Year : Year of entry
Month : Month of entry

Data is load into three different targets:

- A relational database: PostgreSQL
- A cloud storage service: AWS S3
- A data warehouse: Clickhouse



Milestones

- **21 Nov 2024:** Define YAML structures and build connectors to Postgres, AWS S3, Clickhouse.
- **28 Nov 2024:** Parse YAMLs into and AST and validate. Work on the Transformation Engine starts.
- **05 Dec 2024:** Transformation engine and Data Quality Checks complete.
- **12 Dec 2024:** Integration, Testing, Documentation.

Programming in Scala & Repository Overview

Most of the core functionality for the ETL framework will be implemented in Scala

- **Pipeline Orchestration**: Config-based setup to define and manage pipeline stages and dependencies.
- **Data Transformations**: Implementing transformation functions like *filter*, *map*, and *aggregate* for dynamic data processing.
- **Parallelization**: Using *Future/Cats* to enable concurrent execution of tasks for improved efficiency.
- **Data Quality Checks**: Implementing data validation checks such as null, uniqueness, and type consistency.
- **Config Parsing**: Using *Circe* to load and parse YAML configurations into Scala case classes.

Code Outside Scala

- **Configuration Files**: YAML files will be used for ETL job configurations.
- **Prometheus and FluentD Configurations**: Setting up Prometheus for observability may require configuration files (YAML or JSON) outside Scala, to set up monitoring and logging.

GitHub Repository: All code, configuration files, and documentation will be hosted on a GitHub repository, making it easy to track changes and collaborate. (https://github.com/kumarc-rishikesh/scetls)

Acceptance Criteria

Data Accuracy: Data transformation accuracy must be at a minimum of 98%.

Sandbox Performance: Sandbox must perform at 90% of production.

Well structured: Code must run and have 0 runtime errors. Graceful exits of jobs only.

Throughput: Must be able to handle a big dataset(bigger than 250MB) with ease.

Goals

- **An Easy to read system:** A major goal of this project is to implement an easy to read system. People with little technical expertise should be able to have visibility into what goes on in the Data Pipelines. Technical aspects and functions will be adequately abstracted such that there would be little reason to go deep.
- **Ease of Management:** A pipeline in the form of a config enables users to follow best practises like versioning, change reviews, branching etc. This would require only Git - something everyone in the tech space is comfortable with.
- **Utility:** Although YAML is not the best format for writing configs, with the use of plugins and IDE support and adequate validations, YAML could be hard to go wrong with. A Data Engineer would use functions he has written in Scala for transformations and that is all he would have to do.
- **Reliability:** Data Engineering is a job infamous for being unpredictable. Incoming Data is never ideal. We are building a framework that leverages the guarantees of Functional Programming, trying to make this a more predictable process.
- **Migratable to cloud** (Optional)