Chapter 10 - Using APIs in Data Pipelines

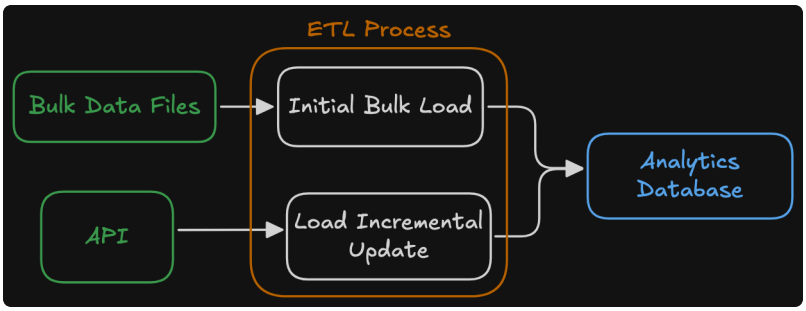
- Structured Processes of Automating the Data = Data pipelines $\xrightarrow{\text{why?}}$ because source data flows into the pipeline and is prepared and stored to create data products $\xrightarrow{\text{other names of this process}}$ Extract, Transform, Load (ETL) OR Extract, Load, Transform (ELT) $\rightarrow$ Depend on the Software Architecture $\rightarrow$ Data Engineer Job $\xrightarrow{\text{Which}}$ is specialized role that focuses on the development and operation of data pipelines $\xrightarrow{\text{but!}}$ It can be done by Data Scientists, Data Analyst or Infrastructure Engineers too!

Types of Data Sources For Data Pipelines

- APIs
  - REST APIs are better suited for incremental updates than full loads $\xrightarrow{\text{why?}}$ because sending the full contents of a data source may require many network calls
  - SOAP & GraphQL is common too
- Bulk Files
  - Large datasets are often share in some type of bulk file that can be downloaded and processed
    - it is a very efficient way to process very large data source
  - CSV & Parquet are common in the field
- Streaming Data and Message Queues
  - For near-real-time update of data $\xrightarrow{\text{streaming sources}}$ Apache Kafka OR AWS Kinesis
- Message Queues
  - RabbitMQ OR AWS SQS $\rightarrow$ Provide asynchronous messaging $\xrightarrow{\text{enable}}$ transactions to be published in a holding location and picked up later by a subscriber
- Direct Database Connections
  - Allow user to get data in its original format
  - More common for sharing data inside organization not outside

Planning Your Data Pipeline

- What we implement = Bulk file for initial load + API to update new records



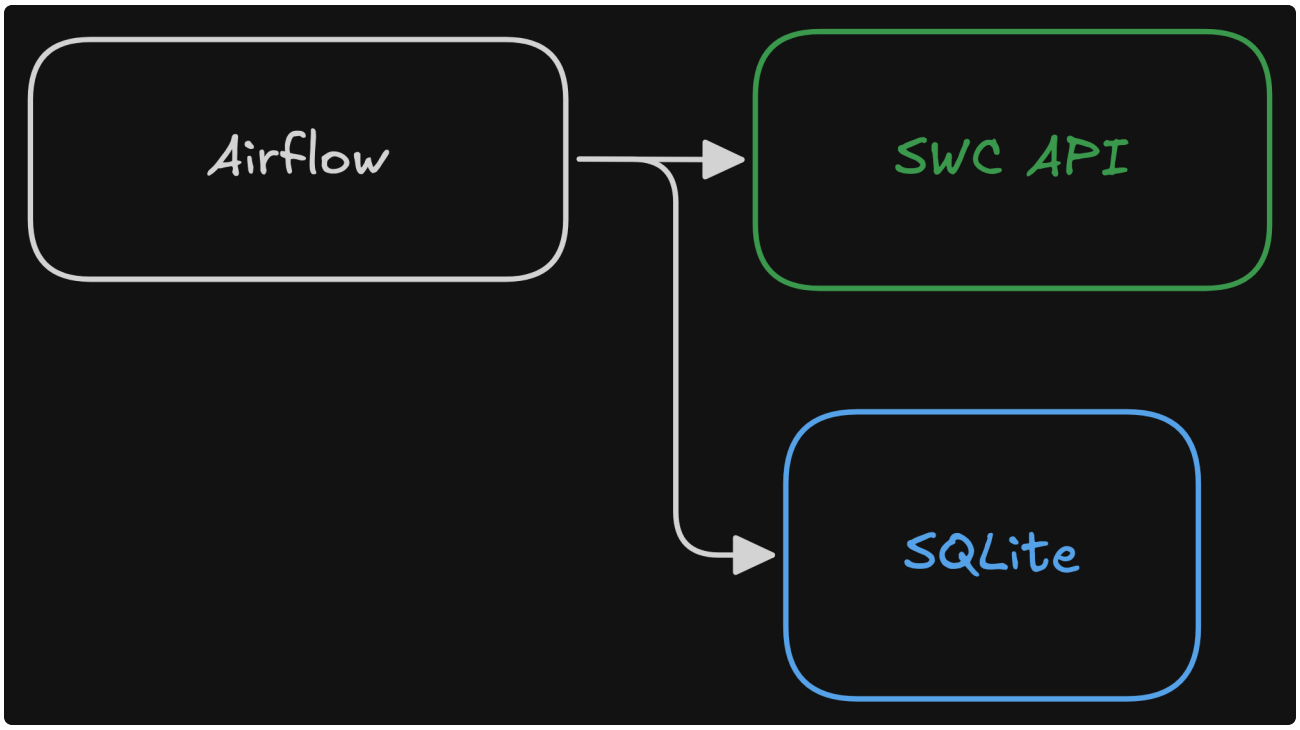
Orchestrating the Data Pipeline with Apache Airflow

> **Harenslak, Data Pipelines with Apache Airflow**
>
> Airflow is best thought of as a spider in a web: it sits in the middle of your data processes and coordinates work happening across the different distributed systems.

Running multiple data processing work streams in production gets complicated quickly. $\xrightarrow{\text{why?}}$ Scheduling, error handling, and restarting failed processes require significant planning and design $\xrightarrow{\text{these processes called}}$ Orchestration $\xrightarrow{\text{who handle it?}}$ Apache Airflow $\xrightarrow{\text{purpose?}}$ instead of coding all the tasks your self use orchestration software $\xrightarrow{\text{what is it?}}$ A full-featured open source engine that uses Python for its configuration + it handles many of recurring tasks involved in data pipelines.

- Airflow have its own specialized terminologies different from other data science programming
  - Check Airflow Glossary
  - Use terminologies from Mathematical Graph Theory:
    - A Node = Process
    - An Edge = Flow between Nodes
    - Directed Acyclic Graph (DAG) = Top-level process that contains steps proceeding in one direction without any loops or recursive logic
      - For each DAG = One Python file
        - Each step in DAG $\overset{called}{=}$ Task
          - Each task will be displayed as a single box on the graph diagram of DAG
          - Operator = predefined template for a task
            - `HttpOperator` will get use to call your API

- `PythonOperator` will get use to update your analytics database
- `XCom` = Cross-Communication: pass information and data between tasks
- Airflow has built-in operators to interact with databases, S3 buckets and other function Check out the Airflow Operators and Hooks Reference



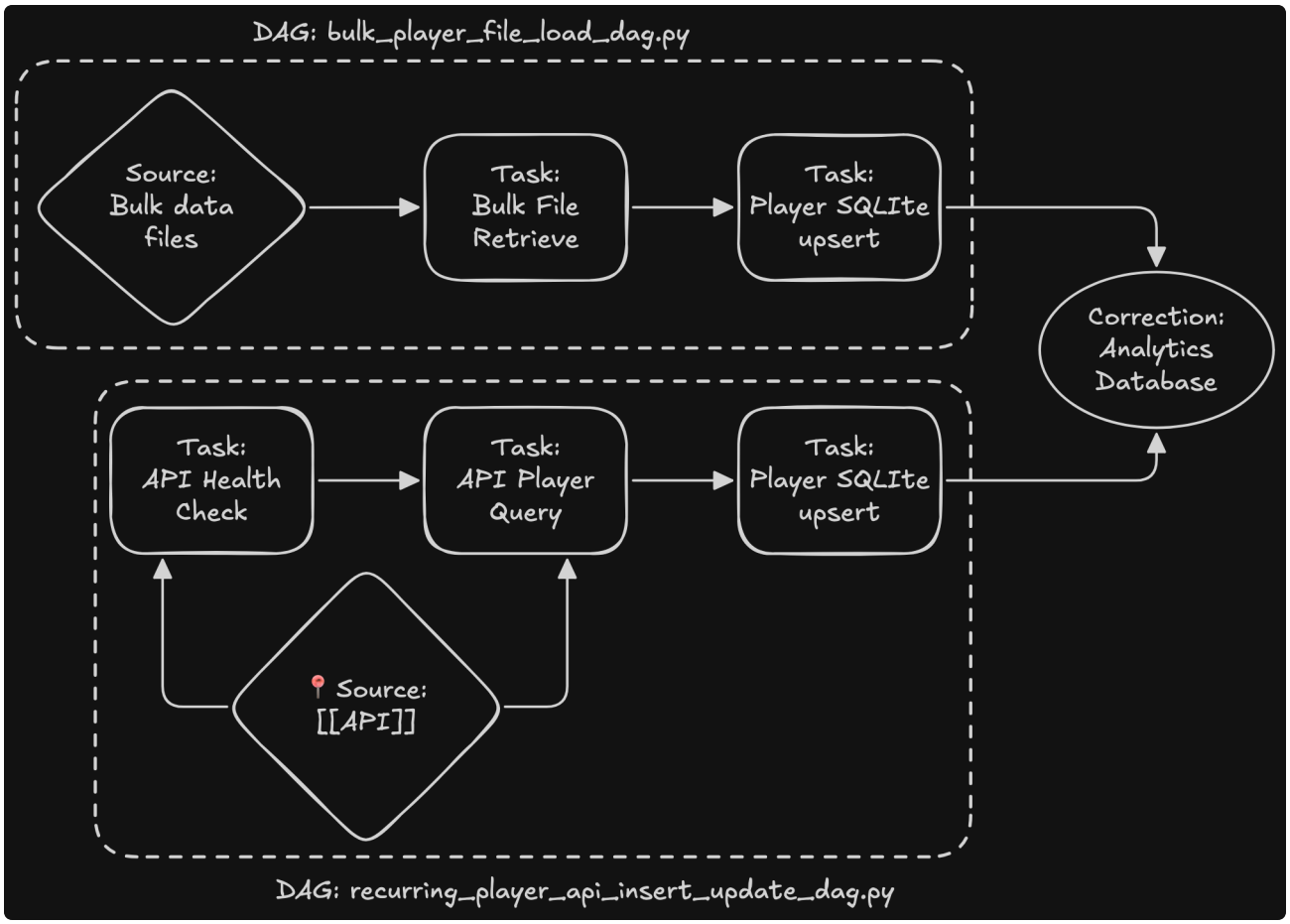
- Running Airflow in Docker — Airflow 3.1.0 Documentation
- Docker Volumes are virtual drives available inside the Docker containers that are mapped to file in your storage
  - They are relative to your airflow project directory
- `docker-compose.override.yaml` to override some of the standard configuration settings within `docker-compose.yaml`
  - it makes troubleshooting easier

Configure Airflow Connections

- Airflow connections allow you to store information about data sources and targets in the server instead of in your code
  - Useful for maintaining separate Airflow environments for development, testing, and production.

    - You will create connections for your API and your analytics database
- Volume mapping

> **You need to install Sqlite yourself**
>
> Sqlite Plugin + `docker exec -it my_container sh`



- `bulk_player_file_load.py` DAG: perform initial load of the analytics database from a bulk file
- `recurring_player_api_insert_update_dag.py` DAG: perform incremental updates of your database using API
    - create these files in dags directory

> **`ipconfig getifadder e0` get your ip address, add this to your host in connections of airflow**

Because you are running airflow on a docker, you can't get access to your api which is running on your machine, so you need to expose it, this nginx - From inside of a Docker container, how do I connect to the localhost of the machine? - Stack Overflow will help you to understand this concept better

⚡ **use `json.loads` NOT `json.load`**

Explanation = python - Why do I get "'str' object has no attribute 'read'" when trying to use `json.load` on a string? - Stack Overflow
`json.loads` : support str and etc.
`json.load` : support text file or binary

Additional Resources:

- Quick Start — Airflow 3.1.0 Documentation
- 🛠️ Getting Started with Apache Airflow 3.x: A Chronological Troubleshooting Guide | by Divesh Kumar chordia | Medium
- UI Overview — Airflow 3.1.0 Documentation
- Templates reference — Airflow 3.1.0 Documentation
- Manage task and task group dependencies in Airflow | Astronomer Docs
- Title Unavailable | Site Unreachable
- `# pyright: ignore`