Team Homework

30 pts Due Jul 27, 2025, 11h59 PM

REAL-TIME FLIGHTS DATA INGESTION CASE STUDY

Assignment Objective

This team assignment is designed to assess your practical understanding of real-world data ingestion challenges and how they are addressed using the Hadoop ecosystem and modern big data tools.

You will work with real-time flight data provided by the AviationStack REST API, which aggregates information from over 10,000 airports and 13,000 airlines globally. Your goal is to build an end-to-end data ingestion and processing pipeline using tools such as Apache NiFi, Hive/Trino, Spark, Kafka, Cassandra, and Superset.

Context

AviationStack offers a RESTful API that delivers structured, JSON-encoded data in real time. Your objective is to:
- Retrieve arrival flights landing at **Montréal - Pierre Elliott Trudeau** Airport (YUL).
- Build a data ingestion pipeline to process, store, and analyze these flights.
- Visualize insights in Superset.

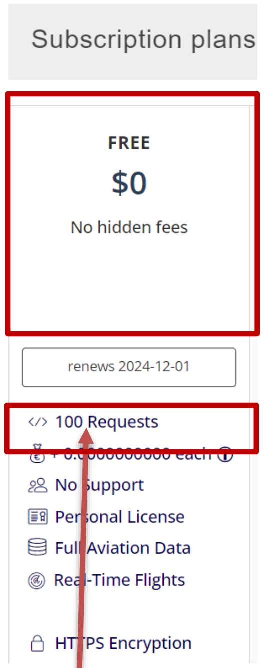
THE CASE STUDY

The Case Study is based on a real project environment that highlights a few challenges often found in Big Data projects. Many of these were addressed as specific topics within the course.

As such you are encouraged, for each course topic, to identify any related information within the Case Study and to use the course lectures / workshops to gain insights into dealing with these challenges.

CASE STUDY REQUIREMENTS

1. aviationstack API allows you to get access to data quickly so you can focus on building the features we need. The data is published through a REST API that returns a JSON-encoded response which contains market stocks data (chart) for a particular stock(s) in a date range.
2. To use any Aviationstack API endpoint, you must pass an **API token** with each request. If you do not include your API token when making an API request, or use one that is incorrect or disabled, Aviationstack returns an error.
3. To obtain or generate your **API Key** Token you need to register (for free) on their website. https://aviationstack.com/signup/free
   For this study, you need to choose the **free plan** for individuals which offer **100 Requests / mo.** Which is enough for the current study.



Subscription plans

FREE

$0

No hidden fees

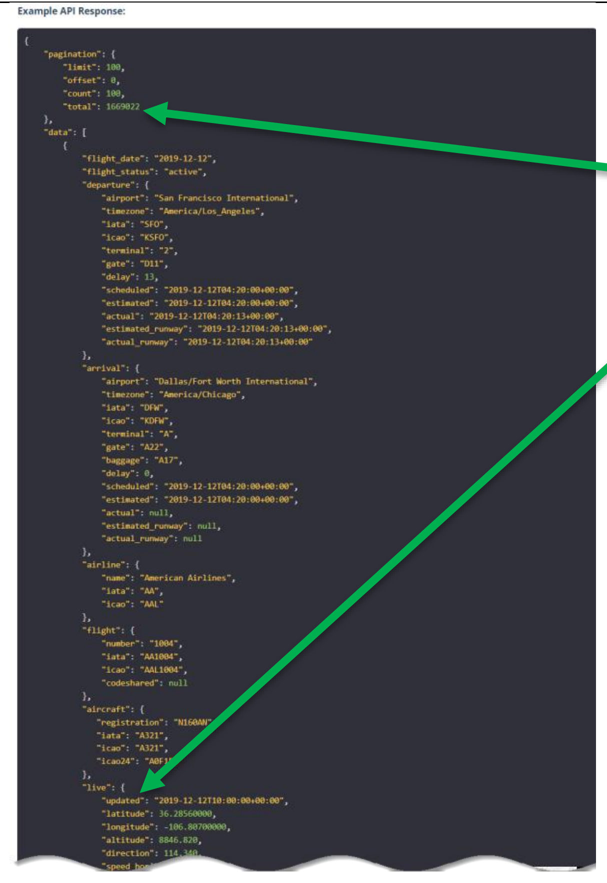
renews 2024-12-01

- 100 Requests
- + 0.0000000000 each
- No Support
- Personal License
- Full Aviation Data
- Real-Time Flights
- HTTPS Encryption

Do not exceed this Limit. Otherwise, you will be charged.

Example API Response

Based on the submitted REST call, the API returns a JSON file structured as follows:

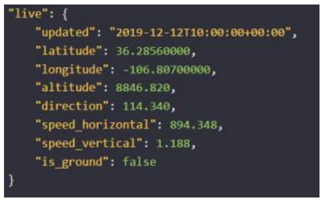


Example API Response:

```
{
    "pagination": {
        "limit": 100,
        "offset": 0,
        "count": 100,
        "total": 1669022
    },
    "data": [
        {
            "flight_date": "2019-12-12",
            "flight_status": "active",
            "departure": {
                "airport": "San Francisco International",
                "timezone": "America/Los_Angeles",
                "iata": "SFO",
                "icao": "KSFO",
                "terminal": "2",
                "gate": "D11",
                "delay": 13,
                "scheduled": "2019-12-12T04:20:00+00:00",
                "estimated": "2019-12-12T04:20:00+00:00",
                "actual": "2019-12-12T04:20:13+00:00",
                "estimated_runway": "2019-12-12T04:20:13+00:00",
                "actual_runway": "2019-12-12T04:20:13+00:00"
            },
            "arrival": {
                "airport": "Dallas/Fort Worth International",
                "timezone": "America/Chicago",
                "iata": "DFW",
                "icao": "KDFW",
                "terminal": "A",
                "gate": "A22",
                "baggage": "A17",
                "delay": 0,
                "scheduled": "2019-12-12T04:20:00+00:00",
                "estimated": "2019-12-12T04:20:00+00:00",
                "actual": null,
                "estimated_runway": null,
                "actual_runway": null
            },
            "airline": {
                "name": "American Airlines",
                "iata": "AA",
                "icao": "AAL"
            },
            "flight": {
                "number": "1004",
                "iata": "AA1004",
                "icao": "AAL1004",
                "codeshared": null
            },
            "aircraft": {
               "registration": "N160AN",
               "iata": "A321",
               "icao": "A321",
               "icao24": "A0F1
            },
            "live": {
                "updated": "2019-12-12T10:00:00+00:00",
                "latitude": 36.28560000,
                "longitude": -106.80700000,
                "altitude": 8846.820,
                "direction": 114.340,
                "speed_hor
```

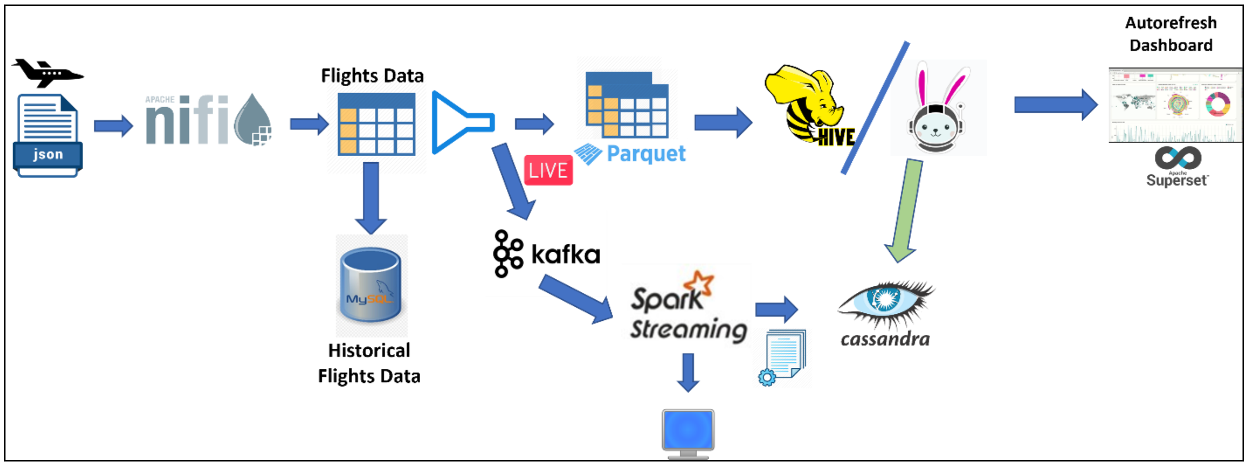
The response contains:

- The total number of elements returned by the REST call.
- The flight data
- Flight live data



```
"live": {
    "updated": "2019-12-12T10:00:00+00:00",
    "latitude": 36.28560000,
    "longitude": -106.80700000,
    "altitude": 8846.820,
    "direction": 114.340,
    "speed_horizontal": 894.348,
    "speed_vertical": 1.188,
    "is_ground": false
}
```

High Level Architecture of the Case Study:



You are asked to:

READING JSON FLIGHTS DATA (5 pts)

- Build an Apache NiFi flow to ingest flights **limited** to **YUL** using the REST API.
- Implement pagination using a loop to increment the offset (do not clone InvokeHTTP processors).

⚠ Limit (in the loop) each **API request to 100 rows**. The free plan is limited to **100 rows** per call.

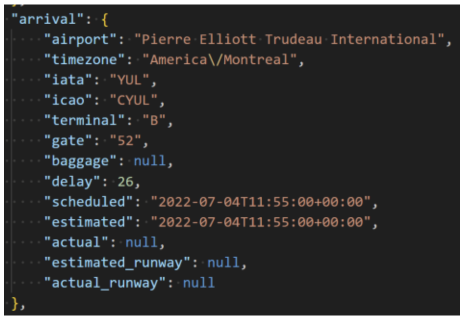
- Avoid hardcoding sensitive or variable information.
- Use saved JSON files during development and testing.

SPLITING/EVALUATING FLIGHTS DATA (4 pts)

- Use Jolt/JSLT to extract **Live** data and send it to a Kafka topic.
- Save the complete dataset to HDFS as compressed Parquet/Snappy files.
- Ensure filenames follow: flights-<timestamp>.parquet and partition the data by year/month/day.

STORING HISTORICAL DATA (3 pts)

Use Jolt/JSLT to extract the **Arrival** section. Store this information in a MySQL table as historical flight data.



```
"arrival": {
    "airport": "Pierre Elliott Trudeau International",
    "timezone": "America\/Montreal",
    "iata": "YUL",
    "icao": "CYUL",
    "terminal": "B",
    "gate": "52",
    "baggage": null,
    "delay": 26,
    "scheduled": "2022-07-04T11:55:00+00:00",
    "estimated": "2022-07-04T11:55:00+00:00",
    "actual": null,
    "estimated_runway": null,
    "actual_runway": null
},
```

DATA ANALYSIS (6 pts)

Create a Hive/Trino table to query the Parquet data. Provide:

- `First 5 rows of data.`
- `Count of flights per airline for the current date.`
- `First and last scheduled flights for the current date.`

CONSUMING REAL TIME DATA (9 pts)

- Use Spark Structured Streaming to read the Kafka topic and output results to Cassandra.
- Use the Haversine Formula (provided) to calculate distance from each plane to YUL.
- Display results on the console (in Zeppelin) and write output to a Cassandra table.

Hint: You might need at least the following columns in Cassandra. (You can add more columns if you need to)

`departure airport, airport iata, airline name, flight iata, arrival estimated`.

DATA VISUALIZATION (3 pts)

Build a Superset dashboard with at least three visualizations. Choose any relevant metrics or KPIs based on the ingested data.

Development Guidelines

- Use GetHTTP/InvokeHTTP in NiFi sparingly to avoid consuming your entire API quota in one call.
- Use process groups to organize your dataflow and avoid hardcoding.
- Save test JSON responses from browser calls and build workflows offline when possible.
- A sample dataset is provided to allow team members to work concurrently.

Note: This case study focuses only on the data ingestion process and not on the flights data analysis.

Meta Information

YUL Airport Coordinates:

`Latitude:`	45.4690
`Longitude:`	-73.7378

Cassandra Configuration:

- Host: localhost:9042
- Keyspace: training
- Auth: None
- Consistency Level: LOCAL_QUORUM

Troubleshooting

Please use carefully the API and look at the requests limit. Otherwise, you will be blocked as you are using the Free Plan.

In case you are blocked, and you need the data to move forward, I can provide you with some data but with **5 pts penalty**.

What to submit:

You must submit the following items:

1. **A Word Document (S2025TH.docx):** *(Only One submission per Group)*
    - Include your **full name**.
    - Paste all **relevant code and results** (limit result outputs to 5–10 rows).
    - Clearly label each task and section.
    - Provide your **comments and explanations** where applicable (in French or English).

2. Your **Nifi Template**.
3. A Copy of your **input data**.
4. A Copy of the **Parquet files**.
5. Your **Zeppelin Notebook(s)** containing the implemented solutions.
6. **An export of your Superset Dashboard**
    - Separated pdf export or jpg inserted into the Word document.
    - Zip export file.
    - Paste your Charts settings (into the Word document).
    - Saved SQL queries if any.

Important Notes

- **Submission Policy:**
    - No submission / withdrawal: **Grade = 0**
    - Late submissions: **No late submissions will be accepted**
    - **No email** submissions allowed.
    - **No** debugging assistance will be provided.
    - Use of AI tools (ChatGPT, DeepSeek, etc.) to generate code: **50% penalty**

Academic Integrity

- Your submission must be your **own individual work**. Any part of the assignment copied from another student or source will be considered **plagiarism**. Penalties include grade deductions, failure in the course, and reporting to scholarship committees and academic authorities.

- Do **not share your assignment** with others. If your work is copied—even without your permission—you may still be held accountable.

Best of Luck !