Here is the comprehensive documentation for **Project 2**.

This project introduces you to **Semi-Structured Data (JSON)**, **REST APIs**, and the power of **Serverless SQL**—three concepts you will use daily as a Data Engineer.

PROJECT 2: THE API COLLECTOR

Role: Data Engineer

Difficulty: Easy (Foundation)

Est. Time: 3-5 Hours

---

1. Executive Summary & Scenario

The Business Problem:

"GlobalShip," a logistics company, is facing delivery delays. They suspect weather is the primary cause but have no historical weather data to prove it.

They need to start archiving weather conditions for their key shipping hubs (e.g., London, New York, Tokyo) every hour.

The Goal:

Build a pipeline that:

1. Connects to an external **REST API** (OpenWeatherMap).
2. Downloads the current weather as a **JSON** file.
3. Saves it into the Data Lake (ADLS Gen2) with a strict folder structure (YYYY/MM/DD).
4. Allows analysts to query this data using **SQL** *without* loading it into a database (using Synapse Serverless).

The Senior-Level Requirement:

- **Zero-Copy Analysis:** We must not duplicate data into a SQL table. We must query the JSON files directly in the lake.
- **Partitioning:** The storage must be organized so that querying "Last Month's Data" doesn't scan the entire lake (Cost Optimization).

---

2. Architecture Overview

We are moving from "Database Storage" to "Data Lake Storage."

The Stack:

- **Source:** OpenWeatherMap API (REST/JSON).
- **Ingestion:** Azure Data Factory (Web Activity or Copy Activity).
- **Storage:** ADLS Gen2 (Hierarchical Namespace).
- **Compute:** Azure Synapse Analytics (Serverless SQL Pool).

---

3. Pre-Requisites (API Setup)

Before designing, you need an API Key.

1. Go to OpenWeatherMap.org.
2. Sign up for a Free Account.
3. Navigate to "My API Keys" and copy your Key.
4. **Test it:** Paste this URL in your browser (replace {YOUR_KEY}):
   https://api.openweathermap.org/data/2.5/weather?q=London&appid={YOUR_KEY}
   - *Success:* You should see a JSON response like {"coord":{...},"weather":[{"main":"Clouds"}...}.

---

4. The Design Phase (Interactive)

As the Architect, you must make 3 critical decisions before building. Write down your answers.

Challenge A: The Storage Strategy (Partitioning)

Scenario: You will run this pipeline every hour for 5 years. That's ~43,000 files.

If you dump them all in one folder weather/data/, your SQL queries will be incredibly slow.

Task: Define the folder path structure.

- *Option 1:* weather/data/2023-10-25-12-00.json
- *Option 2:* weather/year=2023/month=10/day=25/12_00.json
- *Question:* Which one is better for a query like *"Get me all weather data for October 2023"*? Why?

Challenge B: The "Secret" Management

Scenario: Your API Key is like a password. If you hardcode it into the ADF URL (...&appid=12345), anyone with read access to the ADF resource can steal it.

Task: How do we secure this?

- *Hint:* We used this service in Project 1's "Senior Solution."

Challenge C: Schema-on-Read (SQL)

Scenario: You have a JSON file, not a Table.

Task: How do you write a SQL query against a *file*?

- *Question:* Look up the T-SQL command OPENROWSET. What specific "Parser" do we need to specify to read JSON?

🛑 STOP! Attempt the design above before scrolling down.

✅ Architect's Solution (Reference)

Answer A: Partitioning Strategy

We use **Hive-Style Partitioning**:

weather/year=2023/month=10/day=25/city=London/data.json

- *Reason:* This allows the SQL engine to "Prune" partitions. If you query WHERE year=2023 AND month=10, the engine *ignores* all other folders, saving huge amounts of money and time.

Answer B: Security

1. Store the API Key in **Azure Key Vault**.
2. In ADF, use a "Web Activity" to fetch the secret (or reference it directly in the Linked Service if using the REST connector).

Answer C: Schema-on-Read

We use **Synapse Serverless SQL Pool** with OPENROWSET.

SQL

```
SELECT *
FROM OPENROWSET(
    BULK 'weather/year=2023/month=10/*/*.json',
    FORMAT = 'CSV',
    FIELDTERMINATOR ='0x0b', -- Hack to read whole file as one string
    FIELDQUOTE = '0x0b'
```

```
) WITH (doc NVARCHAR(MAX)) AS rows
```

- *Note:* JSON querying in SQL is tricky. We often read the whole file as a single string (doc) and then use JSON_VALUE(doc, '$.main.temp') to parse fields.

---

5. Implementation Guide (Step-by-Step)

Phase 1: Azure Resources

1. **Reuse:** Your ADLS Gen2 account from Project 1.
2. **Create: Azure Synapse Analytics** (Workspace).
    - *Note:* You can create a "Synapse Workspace" without creating a "Dedicated Pool." The "Serverless Pool" (Built-in) comes free (pay-per-TB scanned).
    - *Networking:* Allow "Allow Azure services to access this workspace."

Phase 2: ADF Pipeline (Ingestion)

Step 2.1: The Linked Service (REST)

1. Create a new Linked Service: **REST**.
2. Base URL: https://api.openweathermap.org/data/2.5/weather
3. Authentication: **Anonymous** (We will pass the Key in the query string for simplicity, or add it as a header).

Step 2.2: The Pipeline

1. Create Pipeline: PL_Get_Weather_Data.
2. Add a **Copy Data** Activity.
3. **Source:**
    - Dataset: **REST**.
    - Relative URL: ?q=London&appid={YOUR_API_KEY} (or use dynamic parameters).
    - Request Method: **GET**.
4. **Sink (Destination):**
    - Dataset: **JSON** (ADLS Gen2).
    - File Path: weather/year=2023/month=10/day=25/London.json.
    - *Challenge:* Can you make the date dynamic? Use ADF Expression: @formatDateTime(utcnow(), 'yyyy').

Step 2.3: Dynamic Partitioning (The "Senior" Touch)

Instead of hardcoding "2023", change the Sink Dataset file path to use parameters:

- Directory: weather/year=@{formatDateTime(utcnow(),'yyyy')}/month=@{formatDateTime(utcnow(),'MM')}/day=@{formatDateTime(utcnow(),'dd')}

- Filename: London.json

Phase 3: Analysis (Synapse Serverless)

1. Open **Synapse Studio**.
2. Go to **Data** -> **Linked** -> **Azure Data Lake Storage Gen2**.
3. Navigate to your weather folder.
4. Right-click the JSON file -> **New SQL Script** -> **Select TOP 100 rows**.
5. **Run the query.** You will see the JSON data.

The "Real" SQL Query (Extracting Values):

Paste this into Synapse SQL script to parse the JSON:

SQL

```
SELECT
    JSON_VALUE(doc, '$.name') AS City,
    JSON_VALUE(doc, '$.weather[0].main') AS Weather,
    CAST(JSON_VALUE(doc, '$.main.temp') AS FLOAT) - 273.15 AS Temp_Celsius
FROM
    OPENROWSET(
        BULK 'https://{YOUR_STORAGE}.dfs.core.windows.net/raw-bronze/weather/*/*/*/*.json',
        FORMAT = 'CSV',
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    ) WITH (doc NVARCHAR(MAX)) AS rows
```

---

6. Key Learnings & Takeaways

- **API Ingestion:** APIs are the standard way to get data from SaaS (Salesforce, Workday, etc.). You learned to treat them as a "Source."
- **Partitioning:** Organizing data by Year/Month/Day is the single most important performance optimization in a Data Lake.
- **Serverless SQL:** You didn't CREATE TABLE or INSERT. You queried the raw files directly. This is the **Modern Data Lakehouse** pattern.

---

7. Project Defense (Thesis Questions)

Answer these before moving to Project 3.

Q1. The "Rate Limit" Trap

"OpenWeatherMap Free Tier allows 60 calls/minute. If we want to get weather for 1,000 cities, and we use a 'ForEach' loop in ADF, what will happen? How do we prevent the pipeline from crashing due to '429 Too Many Requests' errors?"

Q2. The "Schema Drift"

"Tomorrow, OpenWeatherMap changes their JSON structure. They rename main.temp to main.temperature. What happens to your Synapse SQL View? Does the pipeline fail, or does the report just show NULL?"

Q3. Security Audit

"We hardcoded the API Key in the URL parameter. Why is this bad even if we use HTTPS? Where does that URL get logged in Azure?"

Q4. Cost Analysis

*"You are using Synapse Serverless. It charges $5 per TB scanned. If you run SELECT * FROM weather (scanning all history) every 5 minutes, will this be expensive? How does your Partitioning strategy help reduce this cost?"*

Q5. Why JSON?

"Why did we save the data as JSON in the lake? Why didn't we convert it to CSV immediately in the Copy Activity? What is the advantage of keeping the 'Raw' format?"

Reply with your answers to the Thesis Questions to complete Project 2.