JIRA Story: *Design and Implement a Cloud-Native Data Pipeline Using Python and Public Data Sources*

Business Objective

The analytics team needs to explore how **environmental indicators** correlate with **economic conditions** using open public data.
Your task is to design, implement, and document a **cloud-first, end-to-end data pipeline** capable of:

- Ingesting two independent external data sources
- Processing and tracking incremental updates
- Storing data in a structured, resilient cloud architecture
- Producing a simple analytical insight from the curated dataset

The architecture, tools, orchestration, and cloud services are entirely your decision.
The sole strict requirements are:

- All ingestion, crawling, and transformation logic **must be implemented in Python**
- **Cloud resources must be used** for storage and execution
- The pipeline must be **incremental**, not just full-refresh

---

Data Sources

1. Public API Source

Option A — OpenWeather Air Pollution API

Documentation:
https://openweathermap.org/api/air-pollution
Data includes pollutants such as PM2.5, PM10, O_3, CO, SO_2, NO_2, etc.
Requires a free API key.

Option B — World Bank Indicators API

Base example:
`https://api.worldbank.org/v2/country/br/indicator/NY.GDP.MKTP.CD?format=json`
Choose any country and indicator.
No key required.

API Ingestion Requirements:

- Must support **incremental ingestion** (pagination, time filtering, deduplication, etc.)
- Must normalize the response into a structured dataset
- Must store raw + processed data in the cloud

---

2. Web Crawler

Use Python to extract a **structured table** from a public website.

Recommended source: Wikipedia
Example: https://en.wikipedia.org/wiki/List_of_countries_by_inflation_rate

Crawler Requirements:

- Extract a full table (inflation, GDP list, demographic list, etc.)
- Clean and normalize column names
- Convert values to consistent types
- Handle at least one HTML inconsistency (missing field, symbol, merged value, footnote, etc.)
- Store raw + processed data in cloud storage

You may choose another public website with a stable table if preferred to be crawled.

Technical Requirements

Your pipeline must satisfy the following requirements:

1. Cloud Usage

You must use a cloud provider (AWS, GCP, Azure, etc.) for:

- Storing raw, processed, and curated layers
- Running ingestion and/or transformation workloads
- Scheduling or triggering the pipeline

---

2. Python Implementation

Python must be used for:

- API ingestion
- Web crawling
- Incremental ingestion logic
- Metadata tracking
- Transformations / curation
- Analytical output

---

3. Incremental Ingestion

Your pipeline must implement an incremental approach:

- Detect new or updated data
- Avoid duplications
- Track ingestion timestamps
- Use pagination or checkpoints where applicable

---

4. Metadata Tracking

Each ingestion run must record at minimum:

- Ingestion timestamp
- Data source identifier
- Row count or processed volume
- Any execution metric of your choice (duration, last successful checkpoint, etc.)

---

5. Curated Dataset

You must produce a **final dataset** that joins:

- The API dataset
- The crawler dataset

This curated dataset must be clean, normalized, and ready for analytical consumption.

---

6. Analytical Output

Provide a **programmatically generated analytical artifact**, such as:

- A chart (PNG/HTML/SVG)
- A summary table produced by your script
- A small Jupyter notebook visualization

Context expectation:
The output should demonstrate that the curated dataset holds meaningful insights.
For example:

- Trend comparison between economic indicator and environmental metric
- Ranking by region
- Numeric summary of correlations

We don't need go crazy here.. the key is that it is **generated by code** and uses the curated dataset.

Deliverables

1. README

Must include:

- **Architecture Overview** (high-level explanation)
- **Chosen Cloud Services & Justification**
- **Incremental Ingestion Strategy**
- **Data Schema Decisions**
- **Instructions to Run the Entire Project**
- **Assumptions & Limitations**

---

2. Final Documentation

This is the main technical deliverable.
It must include:

- **Architectural Diagram (embedded)**
  - Any format is fine (Mermaid, Excalidraw, PNG, Lucidchart, etc.),
    but it must be **included inside this document**, not provided separately.
- **Detailed Pipeline Explanation**
  - Ingestion flow for API and crawler
  - Storage layers (raw → processed → curated)
  - Transformation steps
  - Metadata tracking implementation
  - Orchestration / scheduling design
  - Cloud components used and their responsibilities
- **Data Models / Schemas**
  - Raw structure
  - Processed structure
  - Curated final dataset
  - Explanation of naming conventions and normalization decisions
- **Operational Considerations**
  - Error handling
  - Pagination strategies
  - Retry logic
  - Incremental ingestion approach
  - Idempotency considerations

---

3. Python Codebase

A complete, runnable codebase, including:

- API ingestion module
- Crawler module
- Incremental logic
- Transformation/curation scripts
- Metadata tracking logic
- Analytical output generator

---

4. Cloud Execution Instructions

A short guide describing how the pipeline runs in the cloud:

- Scheduling or trigger mechanism
- Required environment variables or configuration
- Any IAM roles, permissions, buckets, storage paths, or compute resources
- Steps to deploy or execute ingestion/transformation in the cloud

This ensures the pipeline is actually runnable in a cloud environment.

---

5. Final Presentation (Explanatory Walkthrough)

A presentation-style deliverable (slides, Markdown deck, Notion page, PDF, etc.) containing:

- Explanation of the analytical output
- What insight(s) were identified
- How the output was produced from the curated dataset
- High-level architecture summary
- Key technical decisions and trade-offs
- Challenges faced and how they were solved

Acceptance Criteria (AC)

AC1 – Both Ingestion Pipelines Implemented

- API data and crawler data are both ingested
- Raw and processed data stored in the cloud
- Ingestion implements incremental logic

AC2 – Cloud-Based Execution

- At least one part of the pipeline (ingestion or transformation) executes in the cloud
- All datasets reside in cloud storage

AC3 – Transformation + Metadata Tracking

- Data is cleaned and normalized

- Metadata is collected per ingestion run
- Final curated dataset successfully joins both sources

AC4 – Analytical Output

- A programmatically generated chart/table/insight is produced
- Output clearly uses the curated dataset
- Output demonstrates meaningful insight (trend comparison, summary metrics, etc.)

AC5 – Documentation & Presentation Complete

- README provided
- Final technical documentation complete and includes the architectural diagram
- Presentation explaining insights and decisions delivered