**Case Study: Reusable Apache NiFi Template for Ingesting CSV to Delta Lake with Error Funnel & Provenance Tracking**

**Objective**

Design and implement a reusable NiFi data ingestion template that:

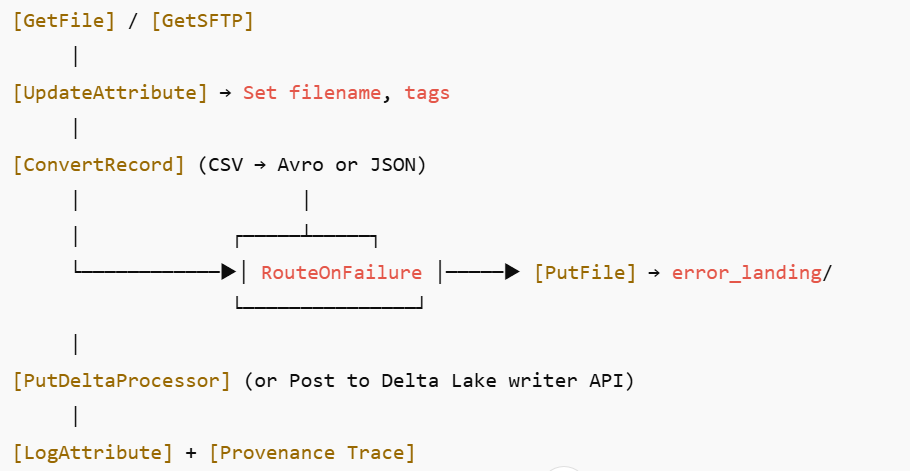
* Ingests CSV files from a source (e.g., SFTP, local, or HTTP).
* Converts them to Delta format for downstream analytics.
* Funnels all processing errors to a centralized "error handling" pipeline.
* Enables full provenance tracking to audit every step of data handling.

**Architecture Overview**

**Key Components:**

1. **Input Source**: CSV via SFTP / Local directory.
2. **Parsing & Validation**: Use schema-aware processors (e.g., ConvertRecord).
3. **Delta Sink**: Output to Delta Lake (typically via Spark or Delta-compatible APIs).
4. **Error Funnel**: Route invalid FlowFiles to a quarantined directory or Kafka topic.
5. **Provenance**: Capture full data lineage using NiFi’s built-in provenance tracking.

**Flow Diagram:**



**Template Components Explained**

**1. Input Processors**

* **GetFile or GetSFTP**: Periodically checks for new CSV files.
* **Properties**:
  + Input directory
  + File pattern filter (e.g., .\*\.csv)
  + Keep source file or archive it

**2. Metadata Enrichment**

* **UpdateAttribute**:
  + Add flow-level metadata: source.name, ingest.timestamp, environment, etc.

**3. Schema-driven Parsing**

* **ConvertRecord**:
  + Input: CSVRecordReader
  + Output: JSON or AvroRecordSetWriter
  + Load schema via controller service (e.g., AvroSchemaRegistry)

**4. Error Handling Funnel**

* **RouteOnAttribute or RouteOnFailure**:
  + Handle schema parsing errors
  + Add reason code via attribute: error.reason, record.count.failed
* **PutFile or PublishKafkaRecord\_2\_0**:
  + Route bad records to: /data/errors/YYYY/MM/DD/ or an error Kafka topic.

**5. Delta Sink (External Integration)**

* Use a custom Spark-based Delta writer OR
* Trigger external job using InvokeHTTP to submit to a Delta Lake ingestion microservice.
* Attach metadata as headers (e.g., dataset name, schema ID).

**6. Provenance**

* Automatically enabled for all processors.
* Use **Provenance UI** to:
  + Replay data from any stage.
  + Track filename, size, and transformations.
  + Audit success/failure at each node.

**Monitoring & Alerts**

* **Bulletin Board**: Enable processor-level error alerts.
* **Back-pressure**: Set thresholds on queues to prevent overflow.
* **Custom DQ Hook**: Optionally use InvokeSlack or PostEmail on failures.

**Test Scenarios**

| **Test Case** | **Expected Outcome** |
| --- | --- |
| Valid CSV with complete data | Ingested to Delta |
| Malformed CSV (e.g., missing column) | Routed to error funnel with metadata |
| File with non-UTF8 encoding | Routed to error, reason: encoding error |
| Delta sink unavailable | Retry mechanism; send alert on failure |
| Large batch (10k+ records) | Batching applied, no memory overflow |

**Reusability Strategy**

**Export Template:**

* Group all processors in a **Process Group**
* Export as .xml template
* Parametrize:
  + Source path
  + Delta table name
  + Schema name
  + Error sink path

**Deploy in Multiple Envs:**

* Use **Parameter Contexts** per environment
* Leverage NiFi Registry for version control

**Best Practices**

* Use **Schema Registry** to decouple logic from structure.
* Avoid inline scripts – prefer Record processors for performance.
* Enforce **naming conventions** for processors and flows.
* Document template as part of metadata lineage using PutDatabaseRecord.

**Tooling Ecosystem Integration**

| **Tool** | **Role** |
| --- | --- |
| Delta Lake | Scalable data lake storage for structured output |
| Kafka (opt) | Real-time error funnel for invalid records |
| Spark | Batch/Streaming Delta ingestion processor |
| Slack/Email | Alert hooks on error or threshold breach |
| NiFi Registry | Versioning of templates for CI/CD |

**Outcome**

This reusable template delivers:

* Consistent, scalable ingestion of structured CSV data.
* Full error traceability and retry capability.
* Transparent data lineage and audit trail.
* Flexibility to evolve schema and logic with minimal rework.