**Case Study: Integrating Great Expectations with Apache NiFi**

**Enhancing Data Quality in Streaming Pipelines: Integrating Great Expectations with Apache NiFi**

**Background**

A data engineering team at a financial services company manages ingestion pipelines using **Apache NiFi**. These pipelines collect customer transaction data from various sources (APIs, CSV uploads, streaming apps). However, the team faced challenges related to **data quality**, especially:

* Missing or malformed records
* Schema drift from upstream systems
* Lack of visibility into validation results

The business mandated tighter controls with automated validation, documentation, and quarantine mechanisms.

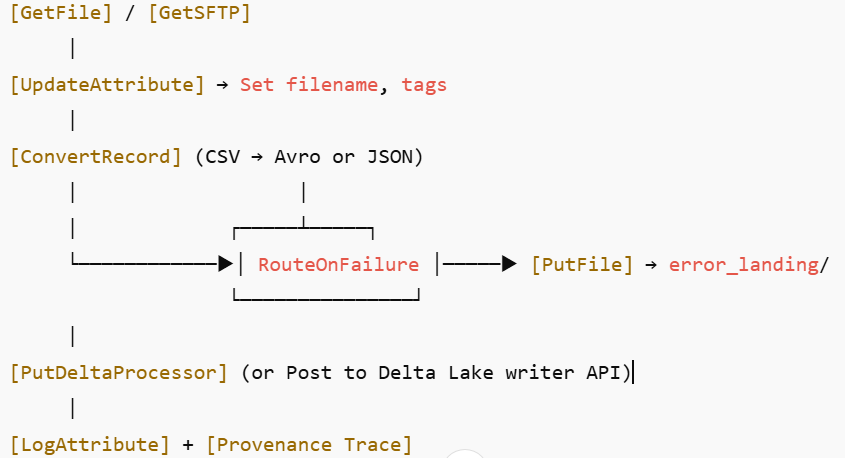
**Objective**

Implement a reusable solution to:

1. **Attach a Great Expectations suite** to key NiFi flows
2. **Publish validation results as HTML Data Docs**
3. **Quarantine bad rows** to a separate stream for inspection or remediation

**Architecture Overview**

* **Apache NiFi** for ingesting and orchestrating data flows
* **Great Expectations (GE)** for validating incoming datasets
* **MinIO / S3** as storage for Data Docs and quarantined records
* **Custom NiFi Processors / ExecuteScript** to call GE validation logic



**Implementation Steps**

**Step 1: Define GE Expectation Suite**

* Dataset: customer\_transactions.csv
* Expectations included:
  + expect\_column\_values\_to\_be\_between("amount", 0, 10000)
  + expect\_column\_values\_to\_not\_be\_null("customer\_id")
  + expect\_column\_values\_to\_match\_regex("email", ".\*@.\*\\..\*")

Suite created using:

great\_expectations suite scaffold transaction\_suite

**Step 2: Package GE as a Python Script**

Create validate\_with\_ge.py:

import sys, pandas as pd

from great\_expectations.core.batch import BatchRequest

from great\_expectations.data\_context import FileDataContext

ctx = FileDataContext("/opt/great\_expectations")

df = pd.read\_csv(sys.argv[1])

result = ctx.run\_validation\_operator(

"action\_list\_operator",

assets\_to\_validate=[ctx.get\_batch(

batch\_request=BatchRequest(

datasource\_name="my\_filesystem\_datasource",

data\_connector\_name="default\_runtime\_data\_connector\_name",

data\_asset\_name="transactions",

runtime\_parameters={"batch\_data": df},

batch\_identifiers={"default\_identifier": "validation\_001"},

),

expectation\_suite\_name="transaction\_suite",

)]

)

exit\_code = 1 if not result['success'] else 0

sys.exit(exit\_code)

**Step 3: Add GE Script to NiFi Flow**

* Use **ExecuteStreamCommand** processor
* Command:

python3 /scripts/validate\_with\_ge.py /data/input.csv

* Success path → next processor
* Failure path → Quarantine

**Step 4: Quarantine Invalid Rows**

* Add a **RouteOnAttribute** or **RouteOnContent** processor
* Filter invalid rows based on GE exit code or flags
* Route to:
  + quarantine/ directory (e.g., in S3/MinIO)
  + Kafka topic for data issue handling

**Step 5: Publish HTML Data Docs**

* Automatically generated with:

great\_expectations docs build

* Upload to:
  + MinIO/S3 bucket (s3://data-docs/)
  + Or serve via NGINX/Apache for stakeholder access

**Results**

* Bad records now automatically separated and reviewed
* Stakeholders use HTML Data Docs to monitor data health
* Validation rules aligned with compliance and reporting standards
* Modular setup allows plug-and-play reuse across other NiFi pipelines