**Enterprise Real-Time Data Platform: Case Study**

**Problem Statement**

A global retail organization with thousands of stores across regions needs a **real-time data platform** that can:

* Ingest **high-velocity POS transaction data** from Kafka.
* Process, validate, and store the data in Delta Lake using **Structured Streaming** and **DLT**.
* Apply **data quality rules** using **Great Expectations**.
* Provide **governance and lineage** through **Unity Catalog and Purview**.
* Allow **secure ingestion from legacy systems** via **Apache NiFi**.
* Meet compliance (e.g., GDPR), support **schema evolution**, **CDC**, and **promote pipelines** across environments (DEV → QA → PROD).

**Skill Tower Developing the Project**

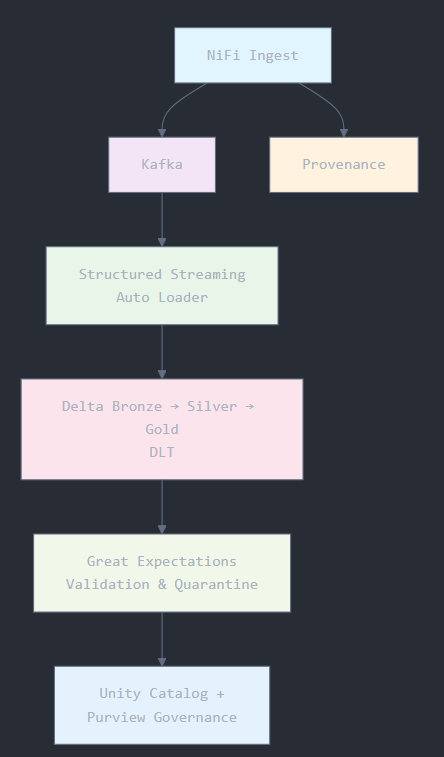
| **Tower Name** | **Responsibilities** |
| --- | --- |
| **Data Engineering** | Kafka, Structured Streaming, Delta Lake, DLT pipelines, Auto Loader, CDC |
| **DevOps & Automation** | DLT YAML management, promotion pipelines, deployment via Terraform |
| **Data Quality** | Great Expectations CLI, rule creation, profiling, checkpoints |
| **Data Governance** | Unity Catalog ACLs, Purview integration, audit logs, GDPR/PII tagging |
| **Platform Engineering** | NiFi ingestion flows, cluster setup, secure Site-to-Site architecture |

**Use Case / Architecture Diagram**

**Use Case Overview**

A hybrid real-time and batch architecture for ingesting, processing, validating, and governing POS transaction data with full lineage and data quality enforcement.

**Architecture Diagram**



**User Stories**

| **Role** | **User Story** |
| --- | --- |
| Data Engineer | As a DE, I want to stream Kafka POS data to Delta with exactly-once guarantee |
| Data Steward | As a DS, I want to profile and quarantine data with quality issues |
| Governance Officer | As a GO, I want to ensure PII fields are tagged and masked |
| DevOps Engineer | As a DevOps, I want to define and deploy DLT pipelines across environments |
| Platform Admin | As a PA, I want secure, scalable ingestion from legacy SFTP/HTTP sources via NiFi |

**Expected Deliverables**

1. **Kafka → Delta Lake** Streaming pipeline using Auto Loader with checkpointing and graceful restart.
2. **DLT Pipelines** (Bronze → Silver → Gold) with schema evolution and CDC support.
3. **Great Expectations** suite with checkpoint scheduling and Slack alerting.
4. **Apache NiFi Flows** with back-pressure, provenance, and version control registry.
5. **Unity Catalog ACLs**, dynamic row filters, column masking UDFs, and lineage integration.
6. **Lineage registration** in Microsoft Purview for source-to-target mapping.
7. **Monitoring hooks**, audit logs to Log Analytics, and promotion scripts (Terraform).

**Milestones & Duration**

| **Milestone** | **Duration** |
| --- | --- |
| Requirement Finalization & Environment Setup | 1 hr |
| NiFi + Kafka + Auto Loader Streaming Setup | 1 hr |
| Delta Live Tables (DLT) with CDC & Promotion | 2 hr |
| Great Expectations Suite + Checkpoint Schedules | 1 hr |
| Unity Catalog + Governance Setup + Lineage | 1 hr |
| End-to-End Testing, Hooking Alerts + Monitoring | 1 hr |
| Final Demo, Documentation, & Handoff | 1 hr |
| **Total Duration** | **8 hr** |

**Implementation Notes**

**Structured Streaming**

* Used **micro-batch** trigger with 10s frequency for stable throughput.
* Watermarks set to 10 minutes for out-of-order tolerance.
* Checkpoints and deduplication ensure **exactly-once** delivery semantics.

**Delta Live Tables (DLT)**

* YAML-based DLT pipelines use CDC merge logic and schema auto-evolution.
* Data quality enforced via EXPECT clauses with fail, drop, quarantine.

**Apache NiFi**

* Ingests legacy CSV/JSON/Avro over SFTP/HTTP using secure Site-to-Site.
* Implements flow provenance and version-controlled templates.

**Great Expectations**

* CLI used to generate rules: null checks, regex, and range rules.
* Quarantined rows stored in separate Delta table with webhook alerts.

**Unity Catalog & Purview**

* Dynamic row filters based on store-region access.
* Column mask UDFs and PII/GDPR tagging enforced.
* Lineage visualized in Unity Explorer; mapped metadata pushed to Purview.

**Evaluation Rubrics**

| **Criterion** | **Max Score** | **Evaluation Criteria** |
| --- | --- | --- |
| **Streaming Pipeline Functionality** | 20 | Working Kafka to Delta stream with checkpoint & watermarking |
| **DLT Pipeline Structure & CDC Logic** | 20 | Proper YAML, CDC merge, promotion across environments |
| **Data Quality Enforcement** | 15 | Great Expectations rules, alerting & quarantine handling |
| **NiFi Pipeline Completeness** | 15 | FlowFile tracking, back-pressure, provenance, template reuse |
| **Governance Configuration** | 15 | Unity Catalog ACLs, UDF masks, row filters, PII tagging |
| **Monitoring & Audit Integration** | 10 | Event hooks, alerting, Log Analytics |
| **Documentation & Diagrams** | 5 | Architecture, lineage, config guides |
| **Presentation & Demo** | 5 | Final walkthrough of all components |