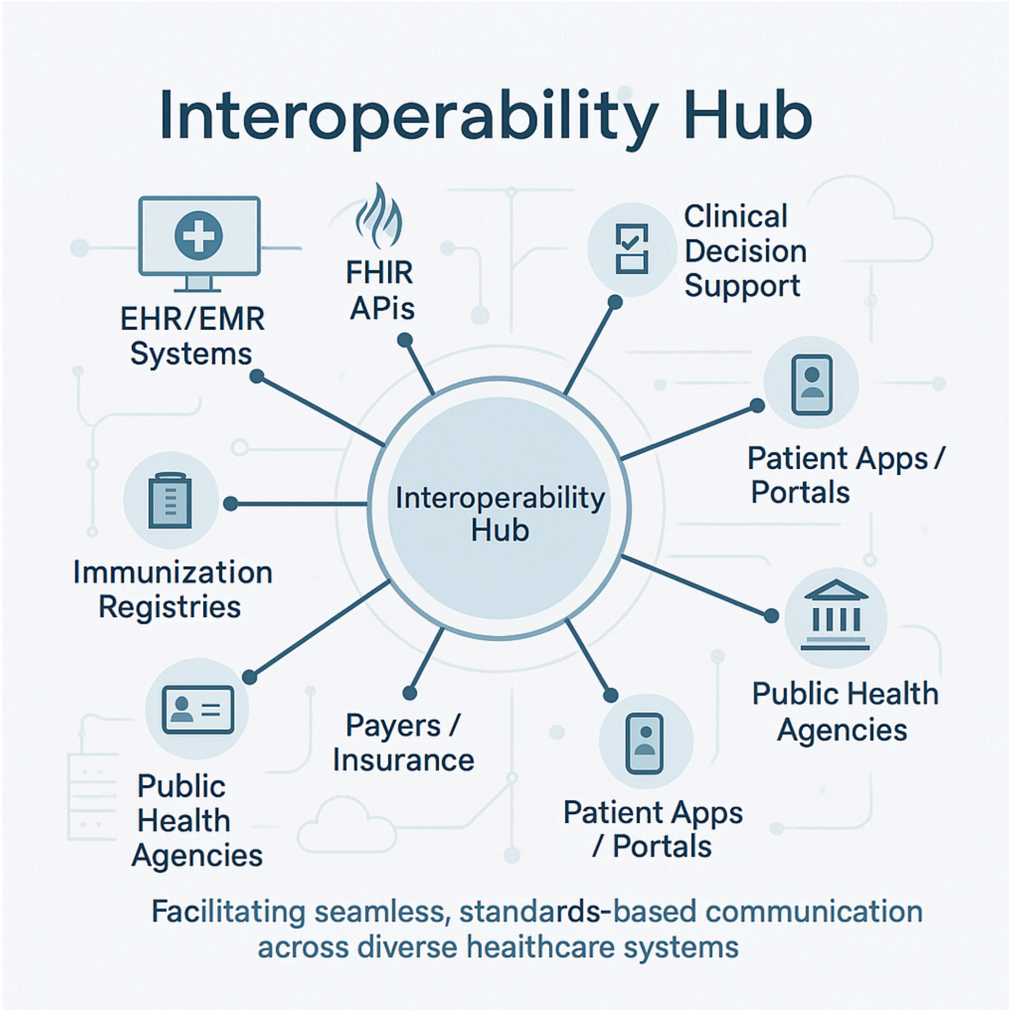
PORTFOLIO

Lowell Buschert

Interoperability Engineer

Ph:1-321-352-9679 Email: lbuschert@gmail.com

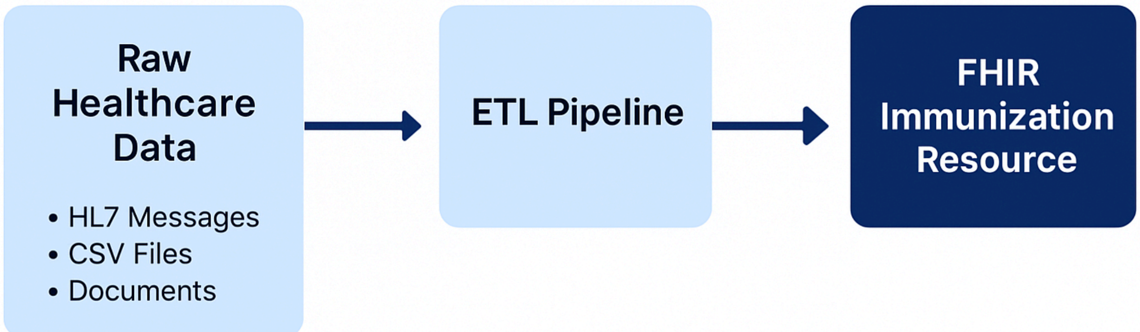


PROBLEM

Healthcare data is often messy — spread across structured, semi-structured, and unstructured sources like HL7 messages, CSVs, PDFs, or database exports. In most clinics, hospitals, or public health agencies, this raw data lacks clear context, making it difficult to use for decision-making or analytics.

Yet this data *has* value — if it can be cleaned, structured, and standardized.

To address this, I developed a prototype Python ETL pipeline that ingests raw healthcare data, applies transformation rules, and outputs FHIR-compliant resources. This provides a clean, analyzable format that can support public health reporting, clinical quality initiatives, and interoperability efforts.



SOLUTION

Python-Based Healthcare Data Integration Pipeline for Standards-Compliant Analytics

This modular ETL (Extract, Transform, Load) pipeline ingests diverse healthcare data sources—HL7v2 messages, flat files (CSV, JSON, XML), or API feeds—and transforms them into FHIR-compliant resources suitable for a wide range of analytics and reporting use cases.

Originally designed for immunization data, the architecture supports extensibility to handle admissions, encounters, procedures, lab results, and population health indicators—making it suitable for hospitals, clinics, dental networks, public health systems, and research institutions.

Key Architectural Features:

- **Modular Design:** Each stage of the pipeline—ingest, transform, load—is independently testable and reusable, allowing rapid adaptation to new data domains or sources.
- **Standards-Driven:** Adheres to HL7v2 and FHIR standards to ensure semantic interoperability with both legacy and modern health IT environments.

- **Centralized Data Dictionary:** A schema-driven mapping approach enables consistent, transparent, and maintainable transformation logic across domains.
- **Analytics-Ready Output:** FHIR resources (Immunization, Patient, Encounter, Observation, etc.) are output in structured form and ready for downstream use in dashboards, reporting systems, or predictive modeling.

Directory Structure Overview:

```
healthcare_pipeline/

├── README.md             ← Project summary, architecture, install
instructions

├── data/                 ← Sample HL7v2, CSV, JSON test files

├── pipeline/

│   ├── ingest.py         ← Reads input from files, folders, or API
endpoints

│   ├── transform.py      ← Maps input data to FHIR resources
(modular by resource type)

│   └── load.py           ← Writes to local storage, databases, or
FHIR endpoints

├── tests/

│   └── test_transform.py  ← Unit tests for transformation logic
across resource types

└── notebooks/

    └── demo_pipeline.ipynb ← Jupyter-based walkthrough for
stakeholders and non-technical users
```

Summary

This pipeline showcases how lightweight Python tooling can be leveraged to create robust, interoperable healthcare data infrastructure, especially in fragmented environments. Its standards-first design and modularity make it suitable for rapid prototyping, public health reporting, or integration with clinical applications.

VALUE

This pipeline supports a wide array of healthcare analytics, public health initiatives, and operational reporting by transforming and standardizing disparate clinical data sources:

- Population Health & Risk Stratification
  Normalizes multi-source clinical data to support identification of at-risk cohorts, chronic disease management, and care coordination across populations.

- Preventive Care & Gaps in Care
  Detects incomplete care interventions (e.g., screenings, vaccinations, follow-ups) using structured event histories—fueling outreach and compliance efforts.

- Programmatic & Regulatory Reporting
  Generates jurisdiction-ready output aligned with CDC, PHAC, and provincial/state requirements for immunization, communicable disease, and chronic condition reporting.

- Operational & Clinical Dashboards
  Feeds normalized datasets into visualization tools or data lakes to support clinical KPIs, patient engagement trends, and site-level operational metrics.

- Predictive & Machine Learning Applications
  Supplies clean, longitudinal data for analytics platforms requiring FHIR-compliant or structured datasets (e.g., for care quality modeling or resource forecasting).

TECH STACK

The pipeline leverages lightweight, proven Python tooling and is deployable in constrained or production environments:

- Core Language: Python 3.10+
- Data Handling: pandas, datetime, csv, json—for parsing and wrangling structured data
- Healthcare Standards Libraries:
  - hl7apy, hl7 (for HL7 v2 parsing and message handling)
  - fhir.resources, FHIR-Parser, or custom mappers (for FHIR R4 modeling)

- Web/API Support: Flask or FastAPI for optional API ingestion endpoints or FHIR server proxying
- Storage Options:
  - Local: SQLite, CSV, JSON

    - Server-grade: PostgreSQL, MySQL, FHIR servers (HAPI, Microsoft FHIR, etc.)

- Visualization / Debugging:
    - Jupyter Notebooks for interactive demos and stakeholder walkthroughs
    - pytest for transformation unit testing
    - Atlassian Loom for videos and presentations

The tech stack is intentionally lightweight, portable, and open source—no licensing overhead and minimal deployment friction.

STANDARDS & REFERENCE MODELS (EXTENDED)

The pipeline leverages healthcare interoperability standards and customizable mappings to meet diverse implementation needs:

- **HL7 v2.x**
  Ingests inbound messages (ADT, ORU, VXU, etc.) from EMRs, registries, or HIEs with robust support for segmentation and OBX parsing.

- **FHIR R4 / US Core / CA Baseline**
  Exports structured, standards-based resources (e.g., Immunization, Observation, Condition) for integration into APIs, repositories, or analytics platforms.

- **CDC / PHAC / SNOMED / LOINC / CVX / MVX (Modular Support)**
  Optional code normalization using curated value sets for vaccines, lab results, diagnoses, encounter types, and clinical observations.

- **Jurisdictional & Custom Mapping Support**
  Extensible to province/state-specific terminologies, care pathways, and schedule logic without hardcoding, enabling portability across regions.

EXAMPLES

Sample Main Module for pipeline

```
from config import MYSQL_CONFIG
from Transform import Transform_to_fhir
from db_insert import insert_to_mysql
import logging
from GetData import fetch_raw_data
def run_pipeline():

    logging.info("Starting ETL pipeline...")

    # Step 1: Fetch raw data

    raw_data = fetch_raw_data()

    # Step 2: Transform to FHIR objects

    fhir_data = [Transform_to_fhir(row) for _, row in raw_data.iterrows()]

   # fhir_data = [Transform_to_fhir(record) for record in raw_data]

    # Step 3: Store in MySQL, including blob
    insert_to_mysql(fhir_data,MYSQL_CONFIG)

    logging.info("✅ Pipeline completed.")

if __name__ == "__main__":
   # logging.basicConfig(level=logging.INFO)
    run_pipeline()
```

Sample Raw Data:

1007649,PELOT PHARMACY,831 MANATEE AVE E,,BRADENTON,FL,342081243,,,27.495385,-82.554966,POINT (-82.554966 27.495385),2024-01-25 00:00:00+00:00,true,false,,false,false,,false,false,false,false,false,false,false,false,false,false,false,,

Sample Human Readable Output + JSON

Search Patient Immunization

Patient ID: 1007649 Search

Immunization Summary

```
Patient ID: 1007649
Status: completed
Date of Immunization: 2024-01-25
Vaccine Code: 208 (COVID-19 vaccine, mRNA)
Pavlovid: None
Practitioner/Pharmacy: PELOT_PHARMACY
Category (ResourceType): Immunization
```

Record found.

Raw FHIR JSON


```
{
  "status": "completed",
  "patient": {
    "reference": "Patient/1007649"
  },
  "performer": [
    {
      "actor": {
        "reference": "Practitioner/PELOT_PHARMACY"
      }
    }
  ],
  "vaccineCode": {
    "coding": [
      {
        "code": "208",
        "system": "http://hl7.org/fhir/sid/cvx",
        "display": "COVID-19 vaccine, mRNA"
      }
    ]
  },
  "resourceType": "Immunization",
  "occurrenceDateTime": "2024-01-25"
```