End-to-End Data Pipeline Plan for Telecom XML Processing

# Overview

We are developing a scalable, maintainable, and observable pipeline to process millions of XML files from an S3 bucket in a telecom environment. The processed files are converted into flat JSON using a decoder, transformed, and loaded into target tables. Three architectural scenarios are considered.

# Scenario 1: Parse XML → Load to Deep Tables → Transform to Final Tables

Pipeline Flow:

S3 (raw XML)

|

v

[Parser Pods (Kubernetes)]

|

v

[JSON Output to Deep Tables (S3 or Database)]

|

v

[Transformer Pods (Kubernetes)]

|

v

[Final Transformed Tables (S3 / Database)]

# Scenario 2: Parse XML → Directly Transform → Load Final Tables (Skip Deep Tables)

Pipeline Flow:

S3 (raw XML)

|

v

[Unified Parser + Transformer Pods (Kubernetes)]

|

v

[Final Output Tables (S3 or Database)]

Pros: Lowest latency, fewer moving parts.

Cons: Harder to debug, no intermediate storage.

# Scenario 3: Parse XML → Store Flat JSON → Transform Later (Split Stages, No Deep Tables)

Pipeline Flow:

S3 (raw XML)

|

v

[Parser Pods (Kubernetes)]

|

v

[Raw JSON Files in S3]

|

v

[Transformer Pods]

|

v

[Final Output Tables]

Advantages: Stage-based, reusable JSON.

Disadvantages: More S3 storage, JSON management.

# Decoder File Strategy

Convert Excel decoder to JSON.

Mount it via ConfigMap or Docker COPY.

Use: json.load(open("/app/decoder.json"))

# S3 File Fetching and De-Duplication

1. Maintain metadata DB with file status.

2. List all S3 XML files.

3. Filter already processed ones.

4. Process & update metadata status.

# CI/CD Pipeline (GitHub Actions)

Stages:

1. Build Docker Image

2. Run Unit Tests

3. Push to Container Registry

4. Deploy to Kubernetes

Example GitHub Action:

name: CI Pipeline

on: [push]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Build Docker

run: docker build -t myparser .

- name: Push Image

run: docker push myrepo/myparser

# Repository Structure

telecom-pipeline/

├── parser/

│ ├── parse.py

│ ├── decoder.json

│ └── Dockerfile

├── transformer/

│ ├── transform.py

│ └── Dockerfile

├── k8s/

│ ├── parser-job.yaml

│ ├── transformer-job.yaml

├── metrics/

│ ├── exporter.py

├── ci/

│ └── github-actions.yml

├── requirements.txt

└── README.md

# Grafana & Prometheus Metrics Plan

Metrics to Track:

- files\_parsed\_total

- files\_transformed\_total

- error\_count

- avg\_parse\_time\_seconds

- pending\_file\_count

# Technology Roles & Alternatives

| Technology | Role | Alternatives |

|----------------|-----------------------------------|----------------------------------|

| AWS S3 | File storage | GCS, Azure Blob |

| Python ET | XML Parsing | lxml, Spark-XML |

| Docker | Containerization | Podman, Buildah |

| Kubernetes | Orchestration and scaling | AWS Batch, Argo Workflows |

| PostgreSQL | Metadata tracking | DynamoDB, MongoDB |

| Prometheus | Metrics collection | CloudWatch, Datadog |

| Grafana | Dashboarding | Kibana, Tableau |