High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **Enerlytics - Energy Consumption Analytics Service**. The service is a Spring Boot microservice designed to collect, process, and analyze energy consumption data from smart meters, providing secure analytics endpoints and leveraging asynchronous processing, PostgreSQL, and Docker.

---

1. System Architecture Overview

Architecture Description:
Enerlytics is structured as a containerized Spring Boot microservice. It ingests energy data asynchronously, persists it in PostgreSQL, processes analytics, and exposes secure REST APIs. JWT is used for authentication.

Component	Role/Responsibility
API Gateway/REST API	Receives data ingestion and analytics requests
Asynchronous Processor	Handles background ingestion and processing of data
Data Storage (PostgreSQL)	Stores raw and processed energy consumption data
Analytics Engine	Computes analytics and aggregates on stored data
Security Layer (JWT)	Secures endpoints and manages authentication
Docker Containerization	Packages and deploys the microservice

---

2. Component Interactions

Sequence Step	Interaction Description
1	Client sends energy data or analytics request to REST API
2	REST API authenticates request via JWT
3	Ingestion requests are queued for asynchronous processing
4	Asynchronous Processor validates and stores data in PostgreSQL
5	Analytics Engine queries PostgreSQL for analytics requests
6	Results are returned to the client via the REST API

---

3. Data Flow Overview

Data Flow Step	Source	Destination	Description

Data Ingestion	Smart Meter/Client	REST API	Energy data submitted via secure endpoint
Async Processing	REST API	Processor/DB	Data queued and processed asynchronously
Data Storage	Processor	PostgreSQL	Validated data persisted in database
Analytics Request	Client	REST API	Analytics query submitted
Analytics Processing	REST API	Analytics Engine	Engine fetches and computes analytics
Response Delivery	REST API	Client	Analytics results returned to client

---

4. Technology Stack

Layer/Function	Technology/Framework
Application Framework	Java, Spring Boot
Asynchronous Processing	Spring Boot Async/Queue
Database	PostgreSQL
API Security	JWT (JSON Web Token)
Build/Dependency	Maven
Containerization	Docker

---

5. Scalability, Reliability & Security

- **Scalability:**
  - Asynchronous processing decouples ingestion from analytics, supporting high throughput.
  - Stateless microservice design enables horizontal scaling via Docker containers.
- **Reliability:**
  - Persistent storage in PostgreSQL ensures data durability.
  - Asynchronous queues prevent data loss during spikes.
- **Security:**
  - All endpoints secured with JWT-based authentication.
  - Sensitive data transmitted over HTTPS (assumed in deployment).

End of Document