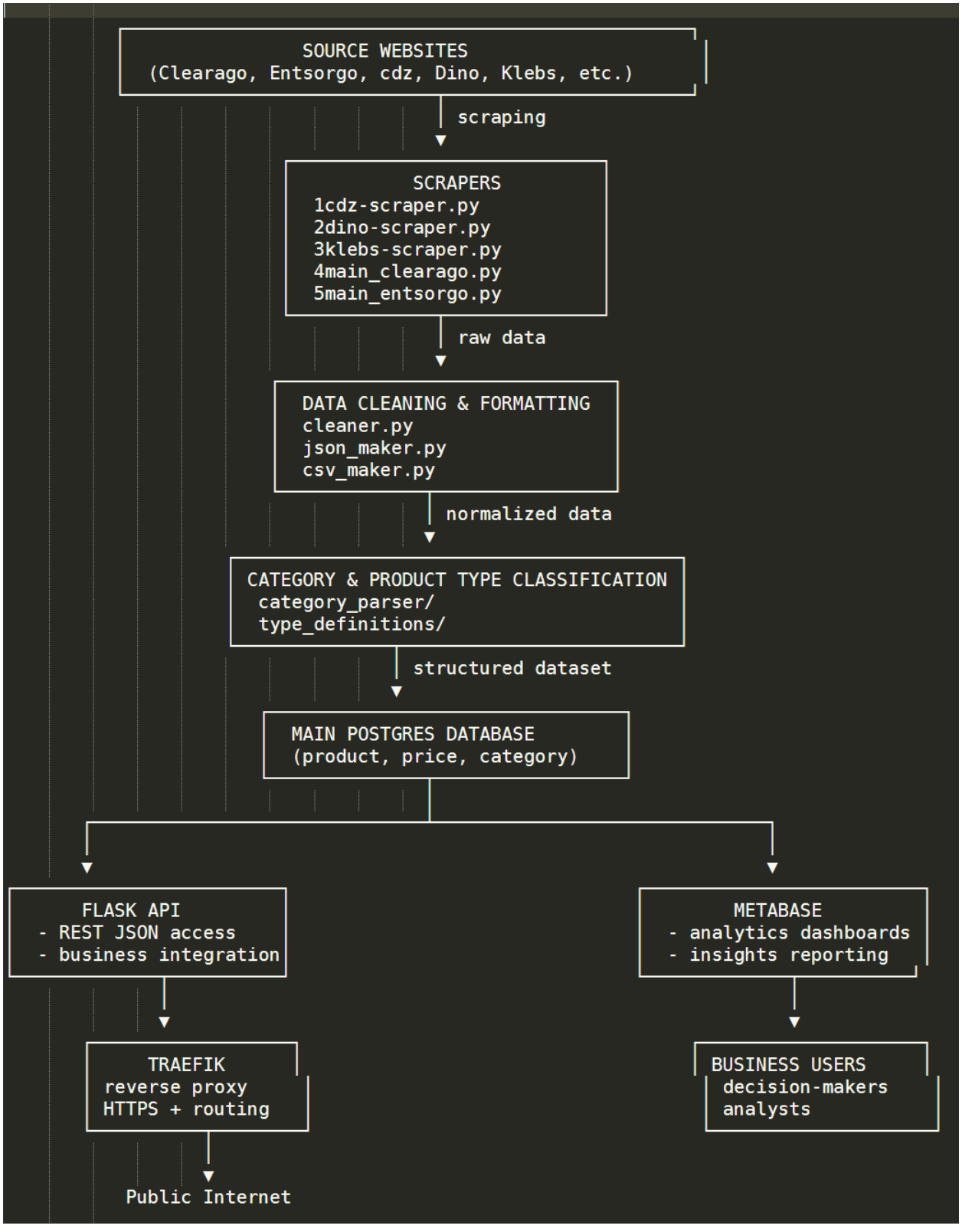
This system automates the extraction, cleaning, classification, storage, and delivery of pricing data from multiple external websites. It is fully containerized, scheduled, monitored, and production-ready, using Docker, Traefik, Postgres, Flask, and Metabase.

Execution Flow — Step-by-Step



A dedicated scheduler container triggers pipeline execution.
Frequency is configurable via environment variables.

1 Automated Scheduling

2 Orchestration

The orchestrator runs runner.py, which launches all scrapers.
Each scraper operates independently for modularity and maintainability.

3 Web Scraping

Multiple Python scrapers extract product and pricing data.
Each scraper targets a different supplier website.

4 Data Cleaning & Standardization

Handles inconsistent units, formatting, missing fields, duplicates, currency, etc.
Produces normalized, reliable intermediate datasets.

5.Category & Product-Type Mapping

Uses rule-based mapping and predefined ontology files.
Ensures harmonized classification across all suppliers.

6 Database Load

Final structured dataset is inserted into the Postgres database.
Data persistence is ensured through Docker volumes.

7 Data Consumption

Flask API provides secure programmatic access.
Metabase provides self-service BI dashboards and visual analytics.

8 External Access & Security

Traefik handles routing, HTTPS certificates, and authentication.
Cleanly separates API access and dashboard traffic.

Core Technologies

Component	Purpose
Python Scrapers	Data acquisition
Docker Compose	Infrastructure orchestration
Scheduler	Automated pipeline triggering
Postgres	Central data warehouse
Flask API	External system integration
Metabase	Visualization & analytics
Traefik	Reverse proxy, SSL, routing

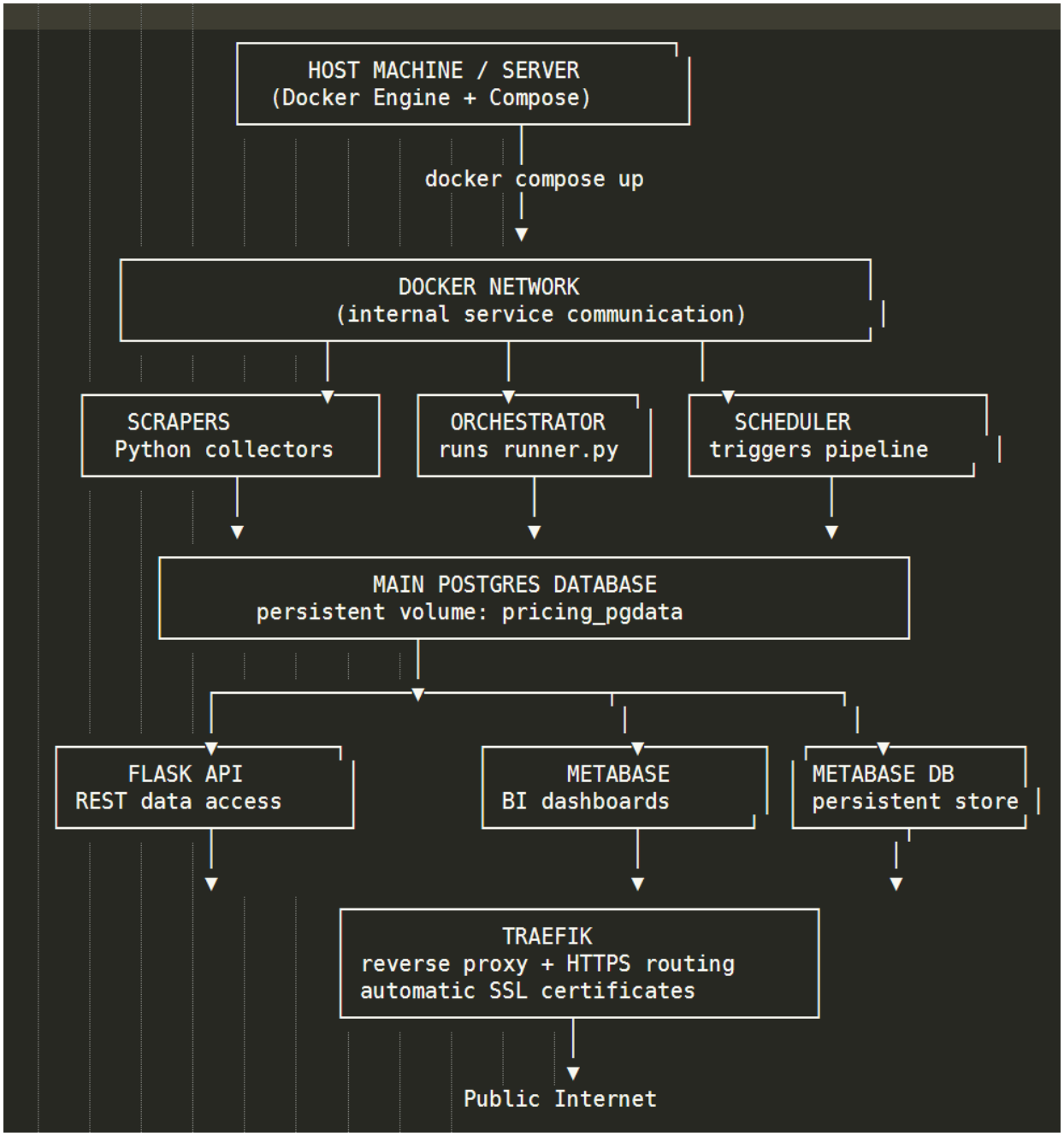
Key Business Value

Fully automated — zero manual effort
Unified product taxonomy across multiple suppliers
Reliable historical pricing intelligence
Secure, scalable, production-ready architecture
Enables analytics, reporting, cost comparison, forecasting
Easily extendable — add new scrapers without redesigning pipeline
Fully Dockerized Deployment

https://www.youtube.com/watch?v=ijqTALUnk1k Milestone 1
https://www.youtube.com/watch?v=iBuDspQvl5s Milestone 2
https://www.youtube.com/watch?v=RJLspqNEm1U Milestone 3
https://www.youtube.com/watch?v=_Zp7e7ik51w Milestone 4
https://www.upwork.com/freelancers/~01207dc9df982f92c4?p=1955092061314666496

If you would like access to the complete project, I can provide it privately upon request. The final delivery was made through GitHub, but the repository remains private to protect client confidentiality.

The entire system runs inside Docker containers, ensuring reproducibility, isolation, scalability, and effortless deployment on any server. Every service — scrapers, orchestrator, scheduler, Postgres databases, Flask API, Metabase, and Traefik — is defined in a single docker-compose.yml file, eliminating manual setup and dependency conflicts. Persistent data, networking, SSL certificates, and scheduled tasks are also fully managed through Docker.



Final Presentation Line

This project is fully Dockerized, portable, and production-ready — deployable with a single command, without installing Python, databases, dependencies, or third-party services on the host machine.