**OPAL**

****

OPAL Open, Procedure, Access, Load

**KI, Agenten, ETL, Docker, Postgres**

Baue mir eine KI die folgendes kann:

ETL Extract -> Transform -> Load

Extract aus mehreren Datensilos bestehend aus

1. Dokumente doc, docx, xls, xlsx, ppt, pptx, txt,

2. Dokumente pdf

3. Grafik jpg, png, … (Inhaltsanalyse?)

4. Musik mit mp3, flac, und den dazugehörigen Tags ….

5. Mails (Adressen, Domains, Personen) aus einer Mail Store DB incl. Anhängen

6. Obsidian Dbs mit .md .png, .jpg, ….

7. Synology Note Station Notizen

8. Zip, RAR, … mit der Verknüpfung zum Transformer unter 1. 2. 3. 4.

9. Sensoren-Messwerte Grad Celsius, Wasserstand,

10. Bücher epub mit allen wichtigen Informationen Autor, Titel, ISBN

11. Maps Karten mit Daten Routen, Tracks, gpx, kml …

12. Gesundheitsdaten HFV, Puls, Blutdruck

13. Archiv Nr, Inhalt als Text

14. ERP Aufträge, Lieferscheine, Rechnungen, Bestellungen,…(PDFs ?)

15….

16….

Die Datensilos bleiben unverändert bestehen und werden nur durch die dazugehörigen Nutzerprogramme verändert. Die Extracts sind dann Inkremental-Extracts, die Extracts können online, 5min, h- und tagesgenau upgedated werden (Block-Chain)

**Folie 1 – Titel**

**ETL mit Docker & Postgres**  
Effiziente Datenpipelines für moderne Anwendungen  
*(Optional: Firmenlogo & Datum)*

**Folie 2 – Was ist ETL?**

**ETL = Extract – Transform – Load**

* **Extract**: Daten aus verschiedenen Quellen lesen (APIs, CSV, JSON, Datenbanken, Verzeichnisse)
* **Transform**: Bereinigen, validieren, anreichern, umstrukturieren
* **Load**: In Zielsystem (z. B. Postgres) schreiben

**Zweck:** Datenintegration & -analyse vereinfachen

**Folie 3 – Warum Docker?**

* **Schnelle Bereitstellung**: Postgres + ETL-Tools als Container starten
* **Reproduzierbarkeit**: Gleiche Umgebung in Dev, Test, Produktion
* **Isolation**: Kein „Dependency-Chaos“ auf dem Host
* **Portabilität**: Läuft überall, wo Docker läuft

**Folie 4 – Postgres als Zielsystem**

* **Open Source, stabil, skalierbar**
* **Starke SQL-Features** (CTEs, JSONB, Window Functions)
* **Einfach in Docker zu starten**

docker run --name etl-postgres \

-e POSTGRES\_PASSWORD=secret \

-p 5432:5432 \

postgres:16

**Folie 5 – Beispiel-ETL mit Docker**

**Setup mit docker-compose.yml**

* **Service 1:** Postgres (Datenbank)
* **Service 2:** ETL-Tool (z. B. Python mit Pandas, Apache Airflow, Meltano)
* **Workflow:**
  1. Datenquelle: CSV-Datei oder API
  2. Transformation in Python (Cleanup, Mapping)
  3. Laden nach Postgres mit psycopg2 oder SQLAlchemy

**Folie 6 – Vorteile dieser Lösung**

* **Schneller Start** ohne komplexe Installation
* **Flexibel** – ETL-Tool kann leicht ausgetauscht werden
* **Skalierbar** – mehrere Pipelines parallel möglich
* **Einheitliche Umgebung** – keine Versionskonflikte

**Folie 7 – Fazit**

* ETL mit Docker & Postgres = **robust, reproduzierbar, flexibel**
* Ideal für **Prototyping, Migrationen, Data Warehouses**
* Nächster Schritt: Automatisierung mit **Airflow / Dagster** und Monitoring