Orchestrierung Open-Data-Aktualisierungen mit Apache Airflow

CAS Data Engineering HS25

Use Case Präsentation

Alexander Güntert

👉 Folien Online oder als PDF herunterladen

Apache Airflow

Stadt Zürich
Open Data

Ausgangsslage / Warum jetzt?

- Stadt Zürich veröffentlicht seit 2012 **Open Data** unter: https://data.stadt-zuerich.ch/
- Open Data Zürich betreibt **zahlreiche historisch gewachsene Aktualisierungsprozesse** über unterschiedliche Kanäle:
  - 📥 Dropzone (WebDAV Harvester)
  - 🤖 GitHub Actions
  - 🦊 GitLab Pipelines
  - ⏰ CRON-Jobs
  - 📤 Manuelle Uploads
- **Zunehmende Anzahl und Komplexität** der Datensätze
- Höhere Anforderungen an Zuverlässigkeit, Monitoring und Nachvollziehbarkeit

➡️ Jetzt ist der richtige Zeitpunkt für eine **Standardisierung und Zentralisierung**

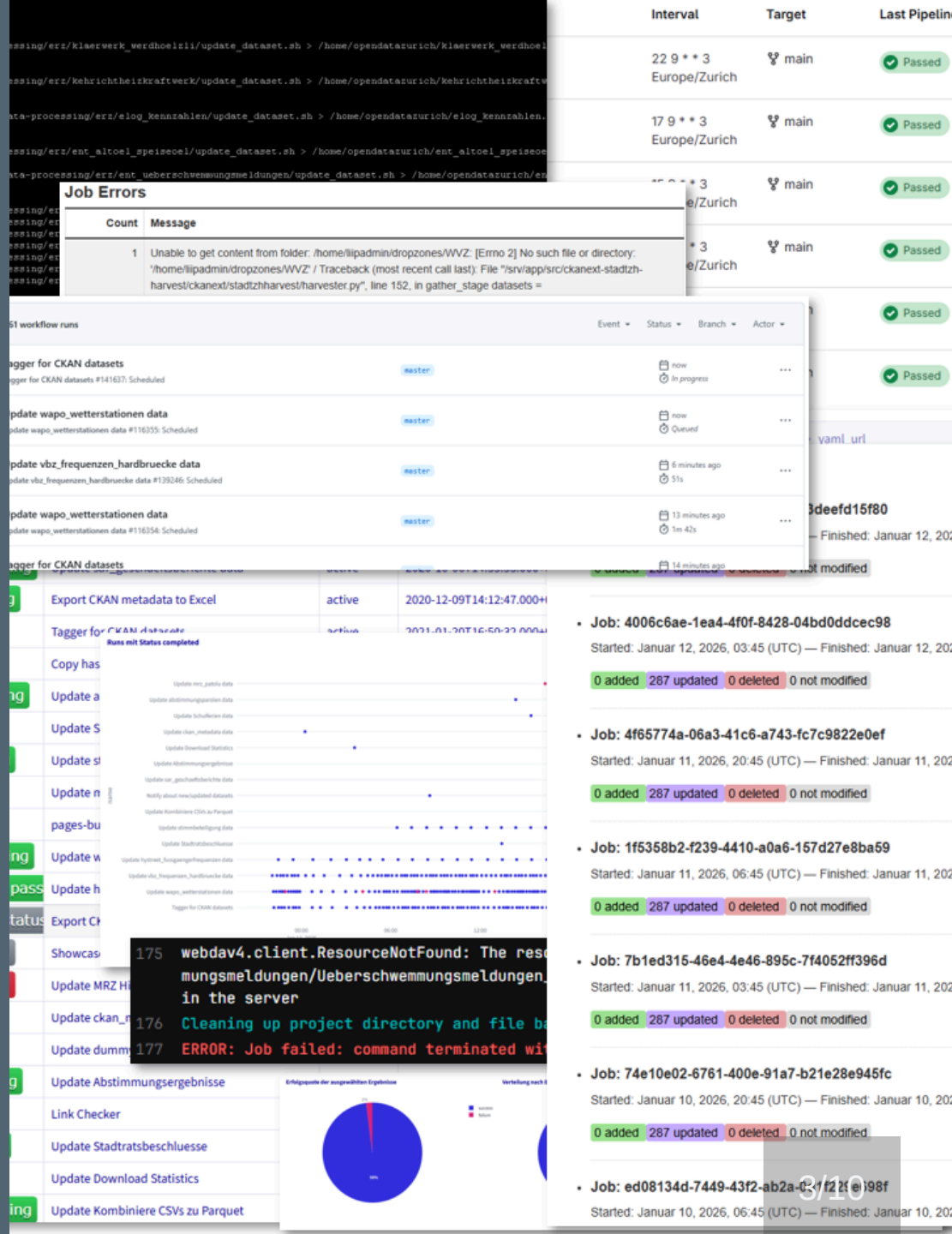
Was passiert, wenn wir nichts tun?

Pain (Ist-Zustand):

- Verschiedene Technologien
- Kein zentrales Monitoring
- Fehler werden oft spät erkannt
- Keine Teilprozesse, kein Retry

Erwarteter Win:

- Bessere Übersicht über alle Prozesse
- Bessere Einbettung in SSZ-Tech-Stack
- Weniger manuelle Eingriffe
- Schnellere Fehlerbehebung
- Höhere Stabilität und Transparenz



Welches Problem lösen wir?

- Fehlende **zentrale Orchestrierung** aller Datenaktualisierungen
- Heterogene Technologiebasis
- Keine einheitliche:
  - Überwachung
  - Fehlerbehandlung
  - Wiederholbarkeit
- Abhängigkeit von **externen Services** (z. B. GitHub Actions)

➡️ Ziel: **Einheitlicher, robuster und skalierbarer Aktualisierungsprozess**

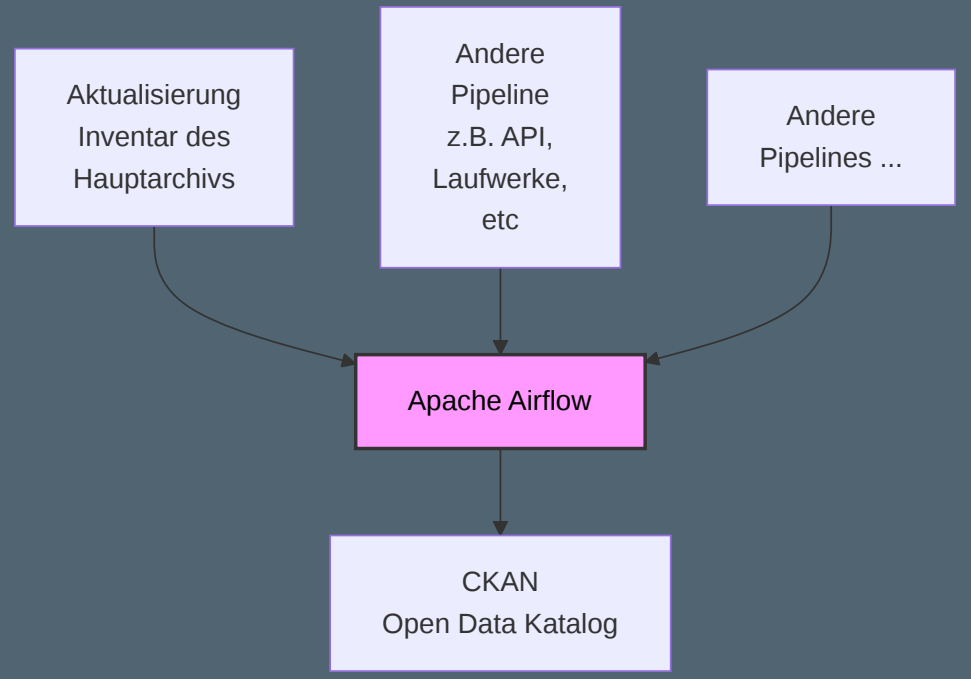
Wie sieht die Lösung aus?

Apache Airflow als zentraler Orchestrator

- Steuerung aller Aktualisierungen über DAGs
- Einheitliches Monitoring & Logging
- Konfigurierbare Retry-Mechanismen
- Klare Trennung:
  - Orchestrierung (Airflow)
  - Fachlogik (Docker-Container ➡️ Sprachunabhängig)

Ergebnis:

- Einheitliche, standardisierte Pipelines
- Bessere Wartbarkeit
- Zukunftssichere Architektur

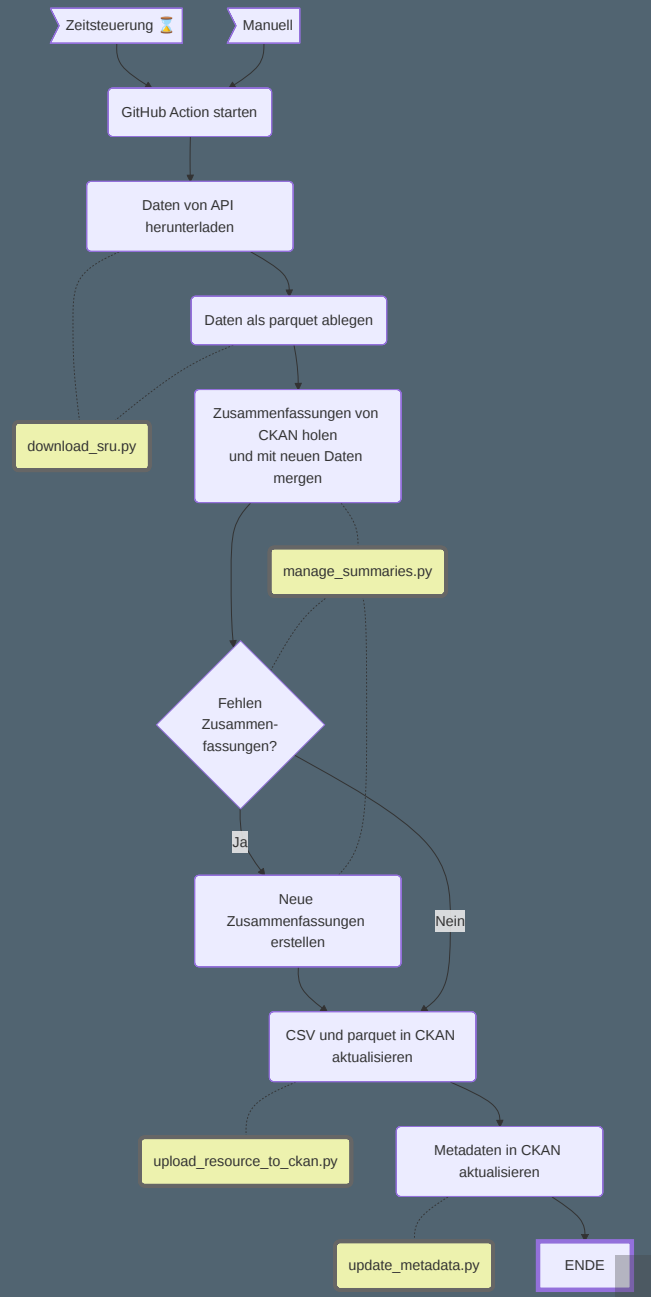


Beispiel-Pipeline: Inventar Hauptarchiv Stadtarchiv Zürich

Warum dieser Use Case?

- Typischer Open-Data-Aktualisierungsprozess
- Hohe Komplexität (API-Abfrage, PDF-Downloads, KI-basierte Textzusammenfassungen, CSV & Parquet, Metadaten-Update in CKAN)

➡️ Ideal als **Blaupause für weitere Pipelines**



Technische Umsetzung (POC)

- Eigener Airflow-DAG für den Use Case
- Verarbeitungsschritte als Docker-Container in separatem Repository
- Gemeinsames Volume für Datenaustausch
- Parallelisierung einzelner Tasks (z. B. CSV & Parquet Upload)

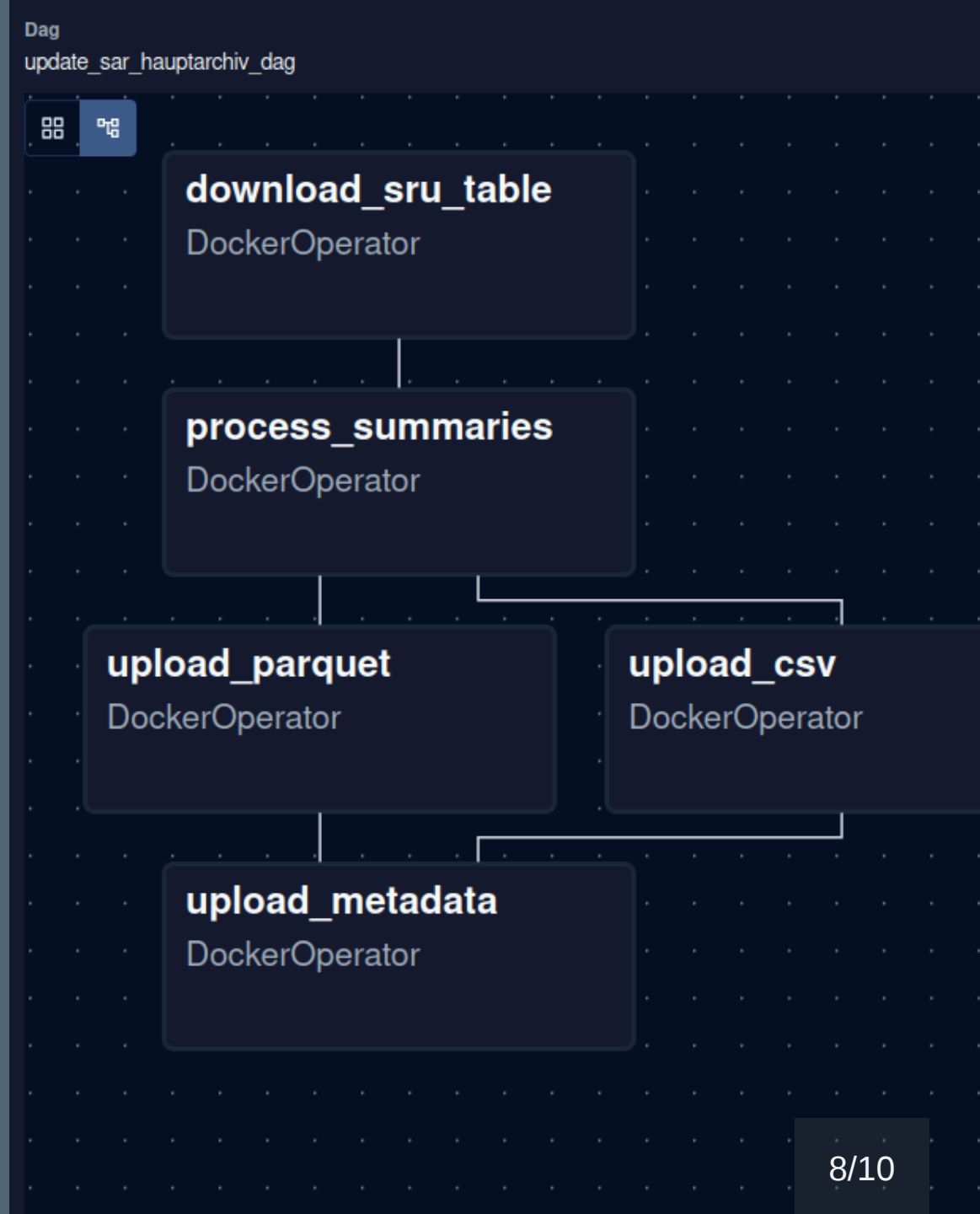
Technologien:

- Apache Airflow
- Python
- Docker Operator
- CKAN API
- Google Gemini API (für Zusammenfassungen)

Was wurde bereits umgesetzt?

- Vollständiger **Proof of Concept**
- Ein funktionsfähiger Airflow-DAG
- Erfolgreiche Aktualisierung:
  - Daten (CSV & Parquet)
  - Metadaten im CKAN
- Dokumentation & reproduzierbares Setup auf Github

➡️ Lösung ist **demonstrierbar und erweiterbar**



Mehrwert der Lösung

- Zusammenführung aller Datenaktualisierungen in einerm Tool
- Flexibilität durch Trennung von Orchestrierung und Fachlogik
- Zentrales Monitoring
- Industriestandard (supported, zukunftssicher)
- Retries oder Teilwiederholungen bei Fehlern
- Unabhängigkeit von Github Actions / Gitlab Pipelines möglich (aber nicht notwendig)
- Einfach erweiterbar durch viele Konnektoren

➡️ **Robuste Basis für zukünftige Open-Data-Prozesse**

Call-to-Action

Nächste Schritte:

- Vorstellung bei stadtinternen Stakeholdern
- Einbindung in SSZ-Toolstack prüfen
- Testdeployment des PoC auf städtischer CMP
- Definition eines Standard-DAG-Templates
- Einbindung zusätzlicher Datenquellen (Externe APIs, Filesysteme)

Perspektivisch:

- Integration des städtischen Metadatenkatalogs (SDK)
- Integration DWH

Bildquelle: ChatGPT