

Orchestrierung Open-Data-Aktualisierungen mit Apache Airflow

CAS Data Engineering HS25

Use Case Präsentation




Alexander Güntert

👉 Folien [Online](#) oder als [PDF](#) herunterladen



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

Terminal output showing directory paths and update scripts:

```
.../erz/klarwerk_verdhoeris/update_dataset.sh > /home/opendatazurich/klarwerk_verdhoeris/...
.../erz/kehrichtheitkraftweck/update_dataset.sh > /home/opendatazurich/kehrichtheitkraftweck/...
.../erz/elog_kennzahlen/update_dataset.sh > /home/opendatazurich/elog_kennzahlen/...
.../erz/ent_altoei_speiseoel/update_dataset.sh > /home/opendatazurich/ent_altoei_speiseoel/...
.../erz/ent_ueberschwemmungsmeldungen/update_dataset.sh > /home/opendatazurich/ent_ueberschwemmungsmeldungen/...
```

Count	Message
1	Unable to get content from folder: /home/lipadmin/dropzones/WVZ: [Ermo 2] No such file or directory: '/home/lipadmin/dropzones/WVZ' / Traceback (most recent call last): File "/srv/app/src/ckanext-stadtzh-arvest/ckanext/stadtzhharvest/harvester.py", line 152, in gather_stage datasets =

Job ID	Status	Started	Finished	Added	Updated	Deleted	Not Modified
4006c6ae-1ea4-4f0f-8428-04bd0ddcec98	Completed	2026-01-12 03:45 (UTC)	2026-01-12 03:45 (UTC)	0	287	0	0
4f65774a-06a3-41c6-a743-fc7c9822e0ef	Completed	2026-01-11 20:45 (UTC)	2026-01-11 20:45 (UTC)	0	287	0	0
1f5358b2-f239-4410-a0a6-157d27e8ba59	Completed	2026-01-11 06:45 (UTC)	2026-01-11 06:45 (UTC)	0	287	0	0
7b1ed315-46e4-4e46-895c-7f4052ff396d	Completed	2026-01-11 03:45 (UTC)	2026-01-11 03:45 (UTC)	0	287	0	0
74e10e02-6761-400e-91a7-b21e28e945fc	Completed	2026-01-10 20:45 (UTC)	2026-01-10 20:45 (UTC)	0	287	0	0
ed08134d-7449-43f2-ab2a-c7f221a198f	Completed	2026-01-10 06:45 (UTC)	2026-01-10 06:45 (UTC)	0	287	0	0

```
175 webdav4.client.ResourceNotFound: The resource
176   /home/opendatazurich/ent_ueberschwemmungsmeldungen/
177   in the server
178 Cleaning up project directory and file base
179 ERROR: Job failed: command terminated with
```

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**

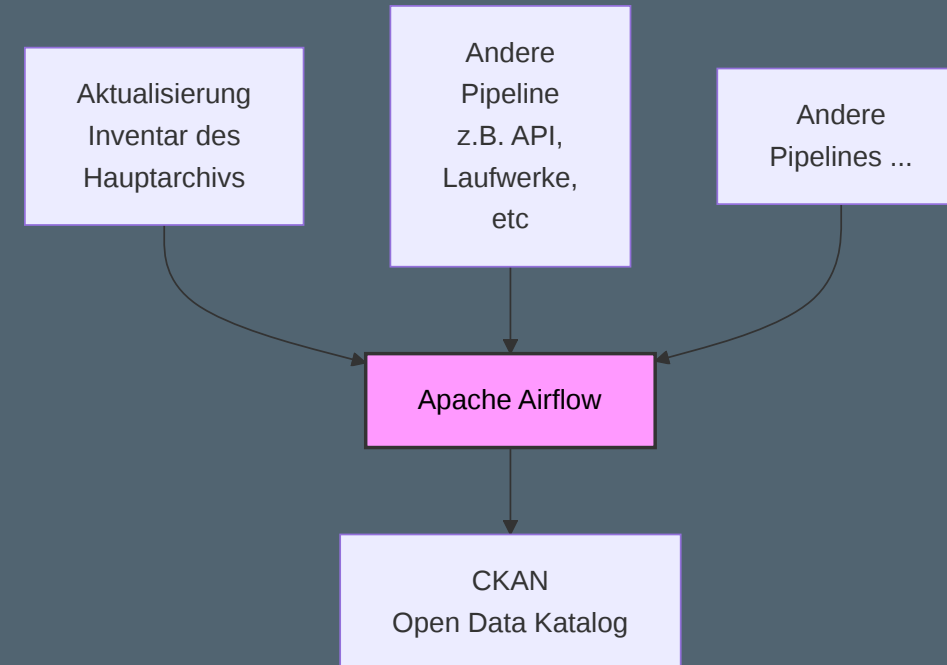
Wie sieht die Lösung aus?

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

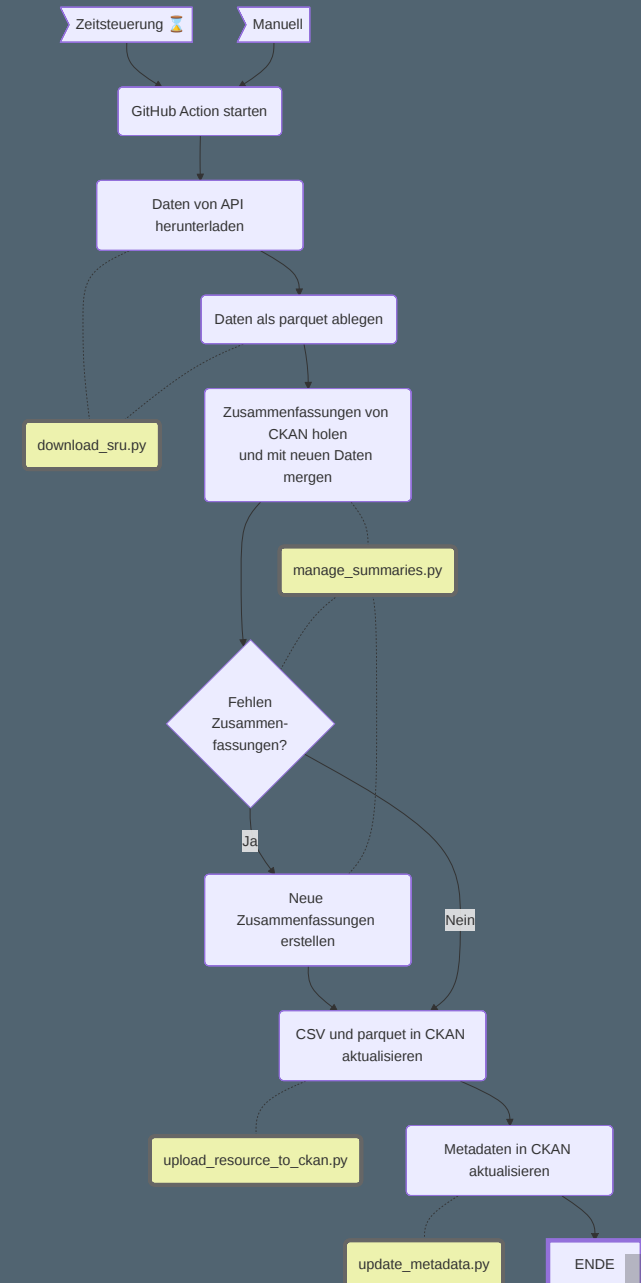


Beispiel-Pipeline: Inventar Hauptarchiv Stadtarchiv Zürich

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:

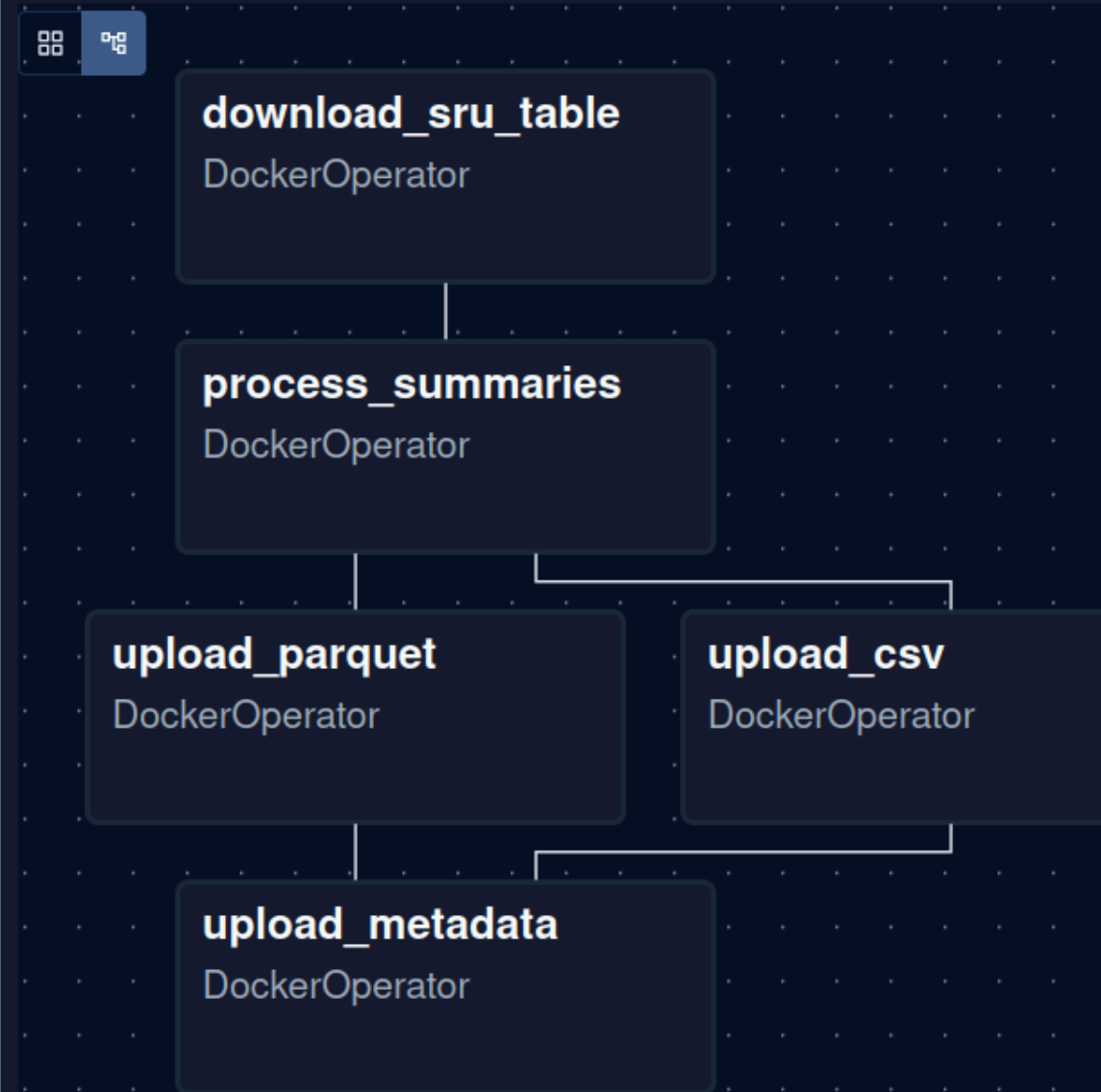
-  Apache Airflow
-  Python
-  Docker Operator
-  CKAN API
-  Google Gemini API (für Zusammenfassungen)

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**

Dag
update_sar_hauptarchiv_dag



Mehrwert der Lösung

- Zusammenführung aller Datenaktualisierungen in einem 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

